Trabajo de fin de Máster
Máster en Diseño Avanzado en Ingeniería Mecánica

Implementación de un algoritmo de transporte público

Autor: Javier Vázquez Peralta

Tutor: Luis Miguel Romero Pérez

Dep. Ingeniería y Ciencia de los Materiales y del Transporte
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 29 de noviembre de 2017

Trabajo de fin de Máster
Máster en Diseño Avanzado en Ingeniería Mecánica

Implementación de un algoritmo de transporte público

Autor:
Javier Vázquez Peralta

Tutor:
Luis Miguel Romero Pérez
Profesor Asociado

Dep. Ingeniería y Ciencia de los Materiales y del Transporte
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 29 de noviembre de 2017

Trabajo de fin de Máster: Implementación de un algoritmo de transporte público

Autor: Javier Vázquez Peralta
Tutor: Luis Miguel Romero Pérez

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Quiero expresar con estas lineas el más profundo de los sentimientos de gratitud y complacencia hacia aquellos 2 que iniciaron este viaje tan lustroso conmigo hace un par de años, cuando éramos unos *chaveas* adentrándose en el mundo ingenieril, el cual se mostró apasionante, severo, indómito y desconocido. Como ese niño pequeño y curioso que mira con sus amigos al mundo que le rodea perpetuamente de una forma diferente y cambiante. Créanme hay personas que aunque no lo sepan, siempre saben, de forma inconsciente, como hacerte cambiar la perspectiva. No conozco mejor forma de experimentar la creatividad, la amistad y la capacidad crítica que con ellos.

A mi tutor Luis, por mostrarse cercano y paciente (virtud muy necesaria conmigo), sabiendo soportar mi carencias sobre Transporte Público, mostrándolo siempre tan amplio como interesante.

Por otro lado, a los amigos (barbacoas nocturnas, charlas en bares, llamadas telefónicas extensas...) que desconocen este campo pero cuyos ánimos, no hacen otra cosa que no sea arrancarte de esta doctrina para darte unas horas buenas de diversión y desconexión, siempre tan necesarias.

La familia siempre es importante por lo que a Raquel, a mis padres, mi hermano, por obvias pero no tanto, horas de aguantar a un hombre bajo presión que necesita desahogarse.

Finalmente, doy gracias por esta bella etapa (junto a las APIs de maps que proporciona Google con su cifrado gratuito que permite a los *mortales* acceder a cierta información oculta que siempre es muy bienvenida). Ha sido un *maldito* placer.

Javier Vázquez Peralta
29 de noviembre de 2017

Resumen

El presente texto explica de forma breve como sustraer a Google Maps información del transporte público (red de autobuses completa) de la ciudad de Sevilla para utilizarla como suministro a la hora de establecer u obtener una solución al problema de asignación de viajeros al transporte público de la ciudad citada. Para modelar el problema, se utiliza la misma solución algorítmica propuesta por Spiess [1]. Como herramientas principales se han utilizado diferentes APIS de Google junto con *MATLAB* y *Mathematica* para hacer posible la resolución del problema, facilitando y haciendo más visuales los resultados obtenidos. Por último, los resultados de la asignación calculada para 164 centroides (dando lugar a 164 rutas óptimas), 1037 paradas de autobús y 10000 enlaces (modo a pie junto a modo autobús) muestran como se distribuye el transporte urbano en Sevilla con unos datos reales, dando una perspectiva global de cuales son las zonas más críticas del transporte público, como influyen los modos a pie, y además, el rol que desempeñan los transbordos posibles incluidos en las base de datos.

Abstract

The aim of this dissertation is to explain briefly how to solve the public transport assignment problem in Sevilla. In order to model that transit network, the same algorithm approach as H. Spiess [1] was empolyed. As main tools of work, Google APIs and commercial softwares as *MATLAB* and *Mathematica* were employed for the development of this work. In fact, they were determinant to show the results clearly in different foms, and obviously, to solve the mentioned problem properly remarking their limitations. Finally, the results of the public transport assignment were determined by Javier's algorithm version implemented in Matlab. Adequate and good results were obtained and displayed to show the capabilities of this software using a *xlsx* database which included all Google Maps data properly organized and structured. Furthermore, the role of public transport transfers and walking paths was also considered in this database in order provide the program with the most complete data input. In summary, the main goals of this project were satifactory accomplished.

Índice

1 Marco del proyecto

Commencer à penser, cést commencer d'être miné

Albert Camus

Los antecedentes de este proyecto, se plantean desde una perspectiva histórica en orden cronológico. No siendo el final otro más que la asignación de pasajeros a los vehículos y el cálculo de las estrategias óptimas para seleccionar el transporte, el cual se ha afrontado desde diferente perspectivas y alcance. Por otro lado, se puede entender, con la consecución de este trabajo, como objetivo la iniciación del alumno a una parte de la ciencia a la que se debe adaptar, dando respuestas concretas, exactas, con visión crítica en base a la bibliografía disponible actualmente. Se proporciona adicionalmente una posible solución a uno de los problemas formulados y propuestos por el departamento con el que se trabaja.

1.1 Introducción

Se pretender dar un breve preámbulo sobre el problema citado previamente. Conforme las ciudades evolucionan han ido incluyendo, creando y modificando su transporte público. La ciudad de Londres fue pionera en este aspecto cuando en 1863 inauguró su primera linea de metro que utilizaba una locomotora de vapor. A esta le sucedieron Nueva York en 1868 , Chicago en 1892, Glasgow y Budapest en 1896, entre otras ciudades que fueron incorporando estas instalaciones para facilitar el desplazamiento de su ciudadanos de un punto de la ciudad a otro, normalmente se conecta el área metropolitana con el centro de la ciudad beneficiando a la polis y sus habitantes mediante este medio de transporte masivo. En aquellos tiempos, las lineas eran reducidas y en el caso de Londres, la primera linea constaba solo de 6 kilómetros de longitud en el denominado *Metropolitan Railway*.

Figura 1.1 Primer metro de Londres restaurado.

Pero conforme el tiempo fue transcurriendo incrementaron su complejidad junto a otros cambios incluyendo autobuses como medio de transporte adicional para el desplazamiento. Hacia 1939 en Chicago, se instauró la

Figura 1.2 Actual vagón del metro de Londres.

primera red de autobuses de *transporte público* junto al concepto de *carril bus*. Su aplicación a nivel regional supuso se estableció cuando se inaugura la conexión Washington-Woodbrige en 1971.

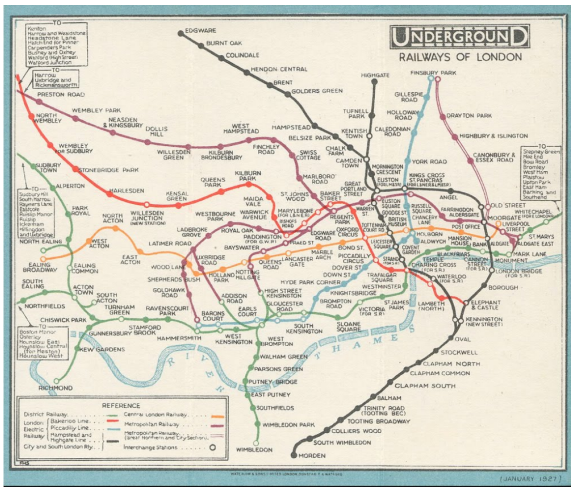

Figura 1.3 Antiguo mapa de lineas de la ciudad de Londres.

Figura 1.4 Mapa actual del metro de Londres.

Con la mejora de las redes de los vehículos utilizados en el transporte bajo el amparo de un código de circulación (que favorece al transporte público), su uso se vio incrementado e incentivado por el poder público del Estado [2]. Además de la capacidades de los vehículos, la velocidad media de los mismos, conllevó un incremento de la frecuencia con la que paraban en las paradas, necesitando un incremento del número de vehículos, mejora de infraestructuras y de los modelos utilizados para describir el comportamiento del transporte público. Esta evolución del transporte puede ser percibida cualitativamente a través de las figuras 1.1 1.2 y 1.5. En lo referente a los mapas utilizados, se puede apreciar la evolución de las mismas observando

1.3 y 1.4.

Figura 1.5 Autobús común utilizando en transporte público.

Ante esta resumida historia surge una pregunta, ¿Cúal es la relación de esta información con el proyecto?

Anteriormente, se mencionó el uso de modelos de transporte para entender mentalmente el funcionamiento del transporte urbano en Sevilla. Sin embargo, aunque este uso es inevitable y recomendable, esta planificación depende de otros elementos: problemas administrativos, el marco institucional, profesionales cualificados y buenos niveles de comunicación con las personas que toman las decisiones, con la prensa y el público son necesarios para que este plan sea efectivo.

La descripción de las principales características de los sistemas de transporte y los problemas asociados es necesaria antes de discutir el enfoque y planeamiento del modelo público de transporte [3].

Congestión en carreteras, retrasos, accidentes, problemas ambientales junto al incremento del tráfico y la demanda de transporte son ejemplos de problemas que afectan a países industriales y desarrollados. Siendo en muchos de los casos poco aceptables porque el crecimiento económico ha superado la capacidad de diseño de estos sistemas preparados para soportar únicamente situaciones que difieren ligeramente de la media. Puesto que estos problemas parecen empeorar por la falta de planificación o su ausencia en ciudades modernas, se requiere de un esfuerzo para maximizar el uso, la capacidad, y en definitiva la eficiencia del transporte público minimizando el coste de estos planes [4].

La demanda de transporte surge derivada de las necesidades que los seres humanos tienen para vivir en sociedad, ya sea para trabajar, por mera diversión o salud. Desde una perspectiva mas abstracta, el viaje de las personas es para realizar una actividad en una localización particular. Por lo tanto, se puede deducir que tiene un carácter cualitativo bastante marcado puesto que depende del día, la fecha, la época del año, entre otros. Sin ajustarse a estos atributos una demanda diferenciada sería inútil. Adicionalmente, se debe tener en cuenta su dependencia del espacio físico por obvias razones como por ejemplo problemas derivados de la coordinación del transporte público que afectarían al equilibrio entre la demanda exigida y el transporte proporcionado previsto. Cabe añadir su irregularidad porque para evitar la congestión en las horas críticas de transporte al trabajo o vuelta del mismo necesitan de una planificación diferente al transporte diseñado para las horas en la tarde [4].

Por otro lado, los medios de transporte proporcionados son un servicio, no pueden ser acumulados, pierden su beneficio si no son consumidos donde y cuando es necesario. Para un buen diseño de transporte sin pérdidas muy acentuadas es necesario adaptar el servicio a su demanda. Para dar este servicio se necesitan infraestructuras y vehículos en conjunto con una reglas de funcionamiento para que el movimiento de pasajeros quede bien en equilibrio con el servicio dado. Complicaciones adicionales surgen si el transporte urbano de una ciudad no es gestionado por una misma entidad, ya que la coordinación se hace mas difícil debido un aumento de la complejidad en la relación entre las diferentes entidades. La congestión del tráfico es también otro efecto a tener en cuenta, puesto que en una ciudad no sólo circula transporte público pero dependen de las infraestructuras de las que posea la ciudad para garantizar su correcto funcionamiento. Un ejemplo, podrían ser los carriles bus garantizando en cierta medida una descongestión del transporte urbano en detrimento del de otro tipo [5].

En términos generales todo esto conduce al *papel* principal que el plan de transporte tiene para solucionar una cierta demanda para personas, en diferentes días y épocas del año, con varios modos de transporte mediante un sistema de transporte con una definida capacidad de operación [6] . Un ejemplo de los elementos que componen a este transporte sería:

- Infraestructuras disponibles. Por ejemplo: la red de carreteras.

- El sistema de gestión de transporte imperante en la ciudad. Por ejemplo: las normas de circulación impuestas para todos los vehículos.
- Modos de transporte y sus operarios.

Desde una perspectiva mas matemática, el modelo de transporte que se utiliza es una herramienta de apoyo cuando se está planificando el propio transporte, no el objetivo. Se utilizan cuando se diseñan y definen las políticas de transporte, cuantificando los resultados y evaluando diferentes alternativas. También pueden servir como instrumentos para obtener los diseños adecuados de ciertos parámetros del sistemas para obtener un óptimo de eficiencia atendiendo a algún criterio por definir [6].

Para el primer caso se estiman variables Y dependientes del sistema en función de las independientes o explicativas, X. Se modela como una función que contienen un vector de parámetros a que habrán sido determinados en la fase de calibración del modelo mediante observaciones del fenómeno. Los valores que corresponden a cada política se diseñan y se comparan las alternativas mediante 1.1.

$$Y = f(a,X) \tag{1.1}$$

La segunda categoría tiene como objetivo encontrar los valores de un conjunto de variables X para obtener un óptimo del sistema modelado a través de la variables Y sujeto a unas restricciones $g(b,X) > 0$. Por lo que se busca maximizar o minimizar 1.1 utilizando las restricciones mencionadas.

Es importante mencionar durante el análisis de un sistema de transporte se deben identificar los elementos que lo componen y sus relaciones. Se puede conseguir siguiendo los siguientes pasos:

- Identificación de las dimensiones espaciales relevantes
- Identificación de las dimensiones temporales relevantes
- Identificación de las componentes relevantes de la demanda de viajes

Teniendo en mente lo escrito previamente, la agregación de los modelos de demanda puede ser de 2 clases (o la medida de los datos):

- Modelos agregados: empleo de datos promedio de las variables a nivel nivel zonal para las diferentes zonas en las que se divide el área de estudio. Al ser medida como promedio el tamaño de la muestra de datos debe ser rico en variedad y generoso en tamaño.
- Modelos desagregados: se considera la demanda de transportes generada de las decisiones tomadas individualmente, en otras palabras, maximización de las preferencias por parte de los usuarios del sistema. Son modelos probabilísticos, estimando la probabilidad de seleccionar cada alternativa, pero no cual se elige exactamente. Estos resultados pueden ser utilizados a nivel de agregación.

Entonces, se llega al modelo de 4 etapas utilizando diferentes fuentes de datos:

- Datos transversales y longitudinales: son la *foto fija* y la *foto móvil*, respectivamente en cuanto a la forma de adquirir los datos.
- Preferencias reveladas y declaradas: reflejan el comportamiento actual y el comportamiento hipotético de los individuos en sus decisiones de viaje, respectivamente.
- Encuestas de hogar realizadas para analizar la opinión de los usuarios mejorar de esta forma la estimación de la demanda.

Inalterado desde los años 60, el modelo clásico de 4 etapas se ha convertido en una práctica común para mejorar las técnicas de modelado en el mundo de la consultoría del trnasporte 1.6. El modelo comienza considerando la zonificación y la red de transporte, la recolección de datos y codificación de la planificación, calibración y validación de resultados. Se consideran los datos de años anteriores que incluyen diversas fuentes. Se utilizan entonces para estimar un modelo del número de viajes generados y atraídos por cada zona de área de estudio, conocida como *Generación - Atracción*. Seguidamente se distribuyen estos viajes en cada una de las concretas zonas, se trata de la fase de *distribución*. El modelo de las decisiones del modo de transporte y sus resultados repartidos entre ellos, es conocida como *Reparto modal*. Finalmente, el último paso es la asignación de los viajes de cada uno de los modos a las correspondientes redes de transporte asociadas (*Asignación*), comúnmente dividida en público y privado.

Situando el trabajo desde una perspectiva tecnológica, el desarrollo de la programación y los ordenadores influye enormemente en cada una de las etapas donde:

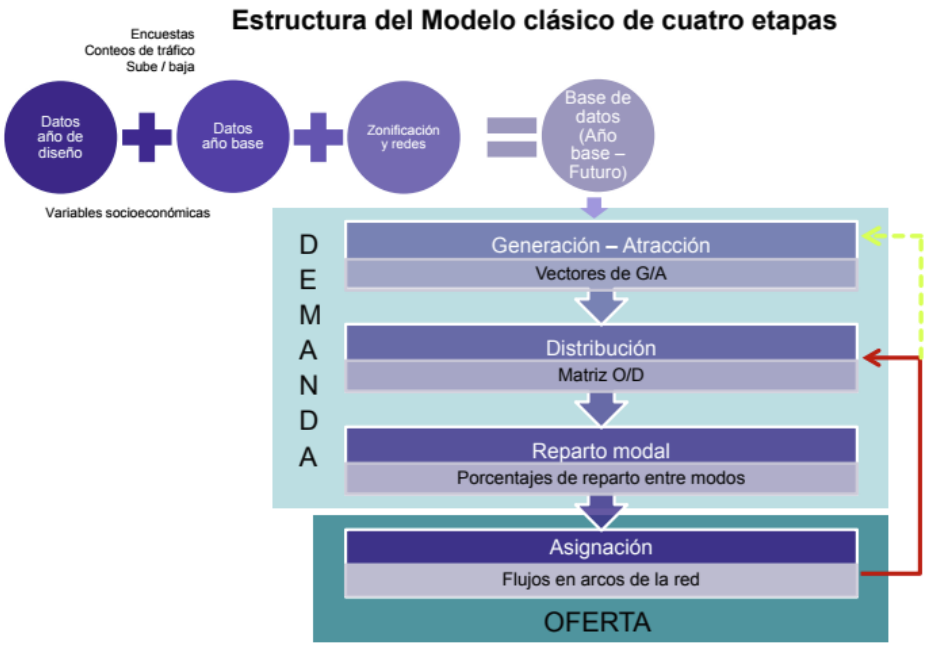

Figura 1.6 Esquema resumen del modelo de 4 etapas.

- Para la *Generación - Atracción* se necesitan los datos de las bases de datos disponibles u obtenibles. Se obtienen de *Google Maps* y de *Tussam*.
- Para la etapa *Distribución*, por medio de *JavaScript* y *Mathematica* se obtiene como están distribuidas las redes de transporte público en Sevilla.
- Etapa de *Reparto modal*, aquí es donde el algoritmo citado [1] modela el comportamiento de los usuarios bajo una serie de hipótesis y asunciones.
- Etapa de *Asignación*, el algoritmo previo también se encarga de hacer un reparto modal de las decisiones tomadas por el volumen de viajeros que viaja de una zona a otra.

Tras la culminación de la tesis doctoral [7] *H. Spiess*, el algoritmo programado en este proyecto fue innovador en el tiempo de su publicación [1]. Además, se implementó en el software comercial *EMME/2* comercializado por *INRO*. En este trabajo, la versión explicada en el artículo ha sido programada y comprobada.

Este trabajo pretende ser novedoso y concreto en la metodología de obtención de datos y resolución, utilizando *Google Maps* y sus *APIs* como fuente para los mismos, incluyendo la posibilidad de obtener la información de cualquier ciudad en el mundo donde no se ofrezcan en abierto. Se organizan de forma innovadora [8], proporcionando una generosa, completa y extensa base de datos sobre la red de transporte de autobús en Sevilla que será utilizado como caso de estudio, aunque ofrece parcialmente la información requerida (incluyendo los transbordos disponibles y enlaces a pie también obtenidos de Google). Posteriormente, se implementa un algoritmo muy conocido en este área [1], la linealización, cuyo fruto serán las soluciones ofrecidas desde un punto de vista crítico, contrastado y transparente en la medida de lo posible.

Desde un punto de vista abstracto, durante el desarrollo del presente proyecto, se ha intentado ser instructivo, por consecuencia descriptivo, con el objetivo principal de que este escrito pueda ser de utilidad para proyectos venideros, para la preparación de clases, facilitando la labor de comprensión del alumnado después de programar y especificar en detalle el funcionamiento de unos algoritmos tan relevantes para la resolución de problemas de transporte público.

1.2 Definición del proyecto

Tras lo expuesto, se pretender resolver la asignación de pasajeros, cuyos datos serán volúmenes de pasajeros reales que utilizan esa red de transporte, mediante los algoritmos encontrados en , donde la fuente de datos

principal de la red de transporte público (lineas, detalles sobre las mismas, mapa de viajes a pie) son Google, y en ocasiones muy concretas, la empresa de transportes de Sevilla, *TUSSAM*. Se buscarán las soluciones mas precisas posibles atendiendo a las hipótesis de partida ya mencionadas. Con el objetivo de tener una perspectiva más crítica, se compararán los resultados de ambas resoluciones junto con resultados obtenidos de otros trabajos similares.

1.2.1 Objetivos principales

Como objetivos principales se toman:

- Entender e implementar la resolución de los algoritmos desarrollados por Spiess [1].
- Diseño de una base de datos eficiente, lógica, compleja y completa sobre la red de transportes de autobús junto a un mapa de accesos a pie de la ciudad de Sevilla.
- Explicar, comprender como funciona el sistema de peticiones de las *APIs de Google* [8]. Ser capaz de formularlas obteniendo los datos que fuesen necesarios hasta completar los campos de la base de datos obtenida. Se dejará como *Bonus track* una completa explicación de carácter informativo e instructivo con el fin de facilitar la labor a alumnos venideros.
- Comprobación de la calidad de la implementación, resolviendo el problema propuesto en el artículo de Spiess mediante su algoritmo. Además, se obtiene la solución para la red mixta de Sevilla previamente mencionada.
- Estudio comparativo entre las soluciones para evaluar la precisión y características de la alternativa utilizada, junto con las consideraciones que surjan de los mismos para futuros problemas y esquemas computacionales que se quieran desarrollar.

1.3 Metodología y herramientas

1.3.1 Metodología

Se presentan varias fases:

- Estudio de conceptos básicos y necesarios de estadística y probabilidad envueltos en el ámbito del transporte público para afrontar el problema y su entendimiento de una forma óptima. Para la consecución de este punto se utiliza el libro *Modelling Transport* ó para mas profundidad [2, 9].
- Comprensión de los algoritmos, desarrollo matemático para fomentar un análisis crítico junto a una mayor profundidad de entendimiento.
- Sustracción de los datos de la red de transporte público a Google mediante las APIs proporcionadas. Desde la lineas de autobús que se necesitan hasta el mapa de enlaces a pie con las paradas teniendo en cuenta los centroides en los cuales está dividido el distrito de Sevilla capital.
- Generar la base de datos mediante la programación de rutinas que permitan almacenar los mismos de forma ordenada y sin errores.
- Escribir las rutinas con los algoritmos pertinentes para la resolución del problema de asignación de volumen de pasajeros dependientes de las rutas óptimas.
- Resolución del problema expuesto en el artículo de Spiess, para tras la comprobación de su idoneidad, resolver el problema real de Sevilla por medio de los algoritmos programados.
- Evaluación: comparativa entre las soluciones obtenida y conclusiones extraídas sobre los resultados arrojados por los programas.

Para aplicar los algoritmos sin problemas que no sean de la lógica de programación se diseña la base datos en primera instancia, y después, teniendo en cuenta la misma se adaptan los algoritmos para acceder a los datos de una forma que no suponga un esfuerzo desmesurado a futuras personas que amplíen o extiendan el proyecto.

Para la consecución de las partes expuestas se utilizaron las herramientas descritas a continuación.

1.3.2 Herramientas

JavaScript Lenguage clave dentro de desarrollo del proyecto, del que se aprendieron únicamente las bases para poder leer comprendiendo las observaciones que Google hacía en sus códigos de ejemplo donde enseña como realizar peticiones mas concretas a los datos que se proporcionan. Durante el proceso de aprendizaje seguido se consultó exhaustivamente las siguientes fuentes [10, 11, 12].

Mathematica Programa principal del proyecto. Se tuvo que aprender casi todo lo utilizado en el proyecto desde 0 puesto que el conocimiento del que se partía era muy reducido. Ha resultado ser un lenguaje extremadamente compacto, útil y agradecido a la hora de mostrar y tratar los resultados de los algoritmos. Gracias al formato *notebook* con el que provee a sus usuarios las temáticas que se trataban y resolvían quedaban en un formato ordenado de apariencia cuaderno de apuntes, de presentación atractiva y claridad inigualable. Se encargó de la creación de las *URL* con las que se realizaban las peticiones, el tratamiento y ordenamiento de la información incluida en las bases de datos, compilador para JAVA permitiendo controlar así *MS Access* para crear las bases de datos. Permite también compilar *JavaScript* y *SQL*, esta última característica es de gran utilidad puesto que organiza la información filtrada desde el lenguaje de *Mathematica* para alojarlo en *MS Access*. Finalmente, permite una gestión y control de las bases de datos mediante *SQL*, y aparte, tiene implementados órdenes propias para el mismo cometido. A continuación se resumen de forma breve cuales son las referencias utilizadas:

- Para consultas generales sobre la sintaxis del lenguaje, estructuras o funciones [13, 14, 15]
- Consultas relacionadas con la gestión de bases de datos [16, 17]
- Consultas relacionadas con la conectividad de *Java* [16, 18]

MATLAB Se implementaron los algoritmos en este lenguaje debido al dominio que el autor tiene sobre el programa para garantizar que fuese eficiente, rápida y segura. También a modo de comparación entre la gestión que ambos programas hacen de las bases de datos que se les proporcionan [19]. Se hizo uso de estructuras de datos tipo *struct* para modelar con facilidad los algoritmos.

Google Maps APIs Para acceder a la información que se ofrece, se utilizaron las siguientes:

- *JavaScript* Se pensó como herramienta para resolver algunos de los problemas que se presentaron durante el proyecto, como la realización de peticiones mas precisas y compleja, el tratamiento de los datos obtenidos incluyendo su filtrado [12].
- *Places (NearbySeach)*. Para comprobar las paradas cerca de los TAZ o centroides en los que se divide el distrito de Sevilla capital [20].
- *Directions*. Tipo de consulta utilizada para averiguar la mayoría de la información relevante de las lineas de transporte público de Sevilla en autobús [21].

Las aplicaciones se pueden utilizar en un modo *demo* pero para acceder a las funcionalidades completas se debe pedir un código a Google (por cada aplicación) que permitirá obtener información mas completa debido a una petición mas sofisticada y controlada por Google. El único problema o límite durante la extracción de datos encontrado es la limitación de peticiones que depende de la aplicación utilizada, del número de peticiones por segundo. Por lo que se tiene la opción de pedir una clave esta viene limitada por un número de peticiones máximas. Se muestra una pantalla de las peticiones disponibles por día que se renuevan a las 9 de la mañana, hora del pacífico en 1.7.

Microsoft Access y Microsoft Excel El uso de ambos es sencillamente como formato donde se almacenan los resultados. Se han puesto ambos porque se generaron bases idénticas para solucionar el posible error que podría ofrecer la conectividad de los drivers que utiliza *Matlab* con respecto a *Mathematica*. Se han utilizado, obviamente, para comprobar que el almacenamiento es correcto desde el propio programa.

SQL Lenguaje utilizado para almacenar, consultar o gestionar bases de datos. Se escogió por lo fácil que es aprender, porque se podía utilizar a través de *Mathematica*, y por el soporte adicional de funciones que el citado tiene para con el lenguaje protagonista de esta sección. Es el encargado de interaccionar con la base de datos y de rellenarla para su uso posterior. En el proceso de aprendizaje se utilizó principalmente [22].

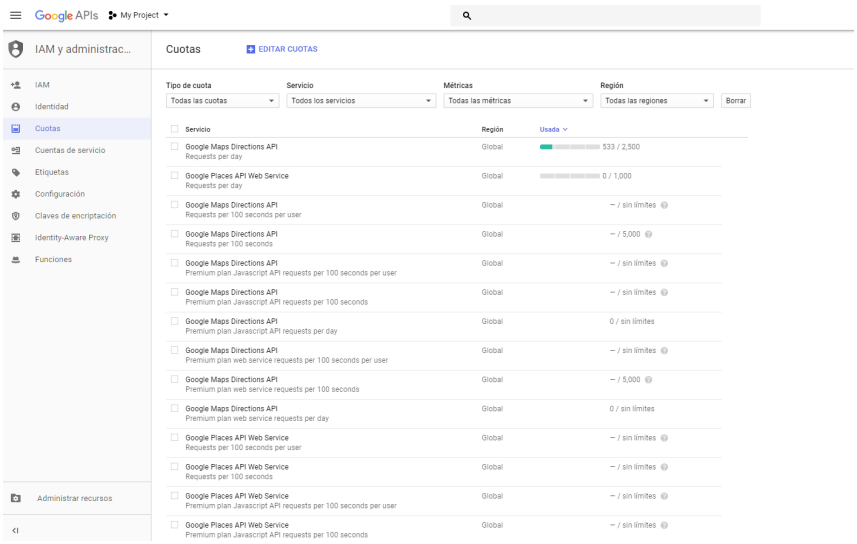

Figura 1.7 Web mostrando las peticiones restantes del día.

Java Se aprendió lo necesario de este lenguaje para el proyecto, de tal forma que permitiese controlar la creación de bases de datos en *MS Access* sin abrirlo ni interaccionar con el programa.

Aplicación Android de *Tussam* Su utilidad principal ha sido como fuente de información contrastada para cotejar y comprobar que la información que se iba obteniendo de las peticiones hechas a *Google Maps* de las diferentes formas como se explicará posteriormente era correcta. Desde la forma del esquema geográfico de las lineas, los nombres de las paradas, la disponibilidad horaria, etcétera. Cada dato ha sido comprobado intensivamente desde esta aplicación junto a *Google Maps* y la aplicación web de *Tussam*.

1.3.3 Criterio de comparación de resultados

Este criterio dependerá del caso resuelto. Para los ejemplos teóricos del artículo [1] la comparación son los resultados del mismo. Idénticamente para los casos de otros artículos mostrados en el capítulo 4.

Sin embargo, para las asignaciones sobre las redes de Sevilla no se tiene un criterio comparativo marcado puesto que no se han encontrado resultados parecidos en bibliografía. Por lo que el sentido común y el entendimiento correcto de los conceptos del modelo de 4 etapas determina las limitaciones, el significado y las mejoras que necesita adoptar el modelo clásico que se pretende resolver.

1.4 Contenido del proyecto

El proyecto queda divido en los siguientes capítulos encargados de la explicación del mismo a un nivel detallado, por fases en secuencia lógicas para una descripción del trabajo sencilla, de sentido común y eficiente.

En el capítulo 2 estado del arte, se define el problema a resolver, con todo el contenido teórico necesario para la explicación del modelo de 4 etapas para modelar el transporte público.

En el capítulo 3 se definen las peticiones, detalles de las mismas, como se realizan, que información proporciona, como se obtiene una clave, que se debe hacer para hacer una petición sin clave. Todo esto junto a los algoritmos de obtención y filtrado de información de estas bases de datos. Se explica como se ha construido la base datos, el esquema de organización de los mismos y el filtrado extensivo que han recibido los datos. Se presentan los resultados obtenidos para las bases de datos.

En el capítulo 4 se definen los algoritmos programados en comparación con otros principales definidos en la literatura remarcando sus diferencias principales y su origen. Al final de este capítulo se presentan los resultados obtenidos para las 2 redes propuestas por *Spiess*.Se presentan los resultados obtenidos usando los datos de Sevilla particularizados en varios casos diferentes y se comentan críticamente.

Por último, en el capítulo 5 se extraen las conclusiones del trabajo planteando lineas futuras a seguir.

2 Estado del Arte

El deseo es una manifestación de toda la vida humana, y aunque en esta manifestación nuestra vida revela a menudo toda su miseria, sigue siendo vida y no la mera extracción de una raíz cuadrada.

Fiódor Mijáilovich Dostoyevski

Se presenta en este capítulo el contenido teórico relevante que soporta y fundamenta el objetivo principal de este proyecto.

2.1 El modelo clásico de transporte estructurado en 4 etapas

Los modelos de asignación de transporte, sea en la versión pública o privada, siguen una serie de normas para alcanzar su cometido. De estas se deducen los objetivos que debería satisfacer un modelo de asignación:

- Primarios
  - Buenas medidas sobre el comportamiento que la red tiene. Por ejemplo: los flujos de una autopista o los beneficios de un servicio de autobuses.
  - Una adecuada estimación de los tiempos de viaje para una demanda dada.
  - La obtención de razonables flujos de transporte e identificar los que estén mas congestionados.
- Secundarios
  - Estimar las rutas utilizadas para cada par origen destino.
  - Analizar cual de estos pares origen destino utiliza una ruta en concreto.
  - Predicciones sobre como afecta a los *links* la introducción de uniones diferentes futuras.

Generalmente, los objetivos primarios deben estar relacionados con los secundarios y la consecución de los primeros debe conllevar automáticamente la de los segundos. Pero comúnmente se consiguen los primeros antes que los segundos porque la visión de los modelos es de conjunto mas que desagregada.

Las entradas básicas de un modelo como este son:

- Una matriz con una estimación de la demanda. Esta se obtendrá de la hora punta del lugar concreto que se quiere estudiar, dando la posibilidad de garantizar otro tipos de estados en el transporte urbano.
- Un red de transporte con sus propiedades.
- Principios de selección de ruta relevantes para el problema a resolver.

Los métodos de asignación de transporte público envuelven una serie de normas que identifican las rutas deseables (las más rápidas cuyo coste generalizado es más bajo) que conectan origen y destino. Entonces, una forma sistemática de localizar viajes origen destino a esas rutas de tal forma que ciertas cualidades de realidad sean alcanzadas.

Los problemas que surgen cuando se modela y se asigna son de muchos tipos; para solucionarlos, *Wilson* proporcionó una larga serie de preguntas a responder por la persona que modela [23]. Estas varían entre el propósito, las técnicas utilizadas. Para desarrollar el modelo presenta ejemplos del enfoque inductivo y deductivo dentro del rol relacional entre los datos y el modelo con el que se trabaje conceptos como complejidad, relevancia de las variables consideradas, interrelación de las misma, entre otros. Como ejemplo de esas preguntas *In how much detail do we need to measure certain variables to replicate a given phenomenon?*.

La especificación del modelo se debe considerar teniendo presentes los siguientes aspectos:

- La estructura del modelo. Ejemplos de preguntas: ¿Cómo afecta la estructura del modelo? ¿Cúal es la más adecuada para resolver el problema?
- El funcionamiento del modelo y su forma funcional. Se refiere a la relación matemática entre las variables, a su peso en el modelo, las simplifiaciones, entre otros.
- Forma de definir las variables, a que hacen referencia, que se encargan de representar.

Después de conocer el modelo en detalle se calibra utilizando datos reales, se validan los resultados y se utiliza. Se puede representar el modelo de forma simplificada como función matemática como la presentada en 2.1. Es importante remarcar que la calibración del modelo y la estimación tienen significados diferentes a lo que se suele conocer. En el primer caso, se requiere de una elección de los parámetros para optimizar la bondad del ajuste de medidas que son función de datos observados. En el caso restante, cuando los resultados que se obtienen no son los adecuados algunos de los parámetros pueden ser determinados como no influyentes siendo apartado del modelo.

$$Y = f(X,\theta) \tag{2.1}$$

Referente a la validación, puesto que los modelos están diseñados con variables de campos muy diferentes hay un tendencia establecida a la interpretación de la bondad de los datos atendiendo solo a los datos proporcionados por datos anuales anteriores. Aunque es necesario, no es suficiente. Se ha demostrado con diferentes modelos mediante cálculos antes y después con resultados realmente dispares. Se requiere de una comparación del modelo validado con datos no utilizados durante este proceso. Si los resultados son aceptables se trata de una confirmación de su validez.

Con el objetivo de entender en detalle cual es la labor del que modela se presenta la figura 2.1. La práctica de estos modelos se utiliza para realizar predicciones condicionales de dos formas:

- Relacionado los valores asignados a las variables política del plan, se evalúa el impacto del modelo.
- En relación con variables asumidas de otras variables

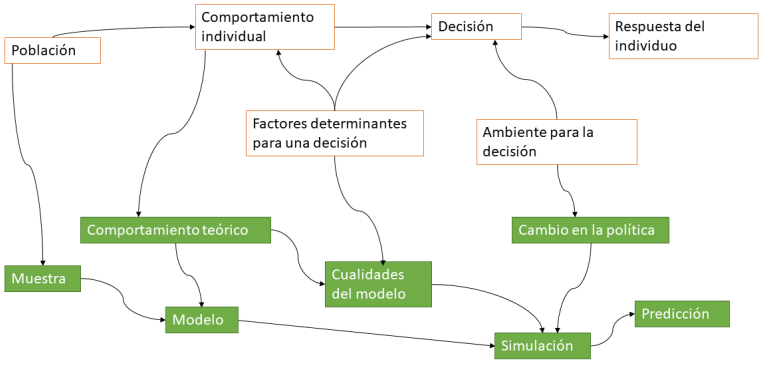

Figura 2.1 El papel de un modelador queda recogido en el color verde de los cuadros dentro de las variables representadas. Fuente [2] .

Se debe recordar a los lectores que hay una diferencia fundamental y sutil entre *modelado* y *predicción*. Con la primera, la persona encargada de modelar debe construir y aplicar el modelo de forma apropiada ayudándose de las herramientas que tenga, las cuales serán sensibles a diferentes factores como las claves de

la política o la lógica de la respuesta obtenida. Sin embargo para el caso 2, es mas importante la precisión y el fundamento de las predicciones así como la capacidad para predecir.

Referente a modelos agregados o desagregados, el nivel que presente el modelo que se diseñe es un importante factor a tener en cuenta. Los agregados se utilizaron en estudios de transporte hasta después de los años 70, son criticados severamente por su inflexibilidad, falta de precisión y coste. Para los desagregados, su popularidad se incrementó en los años 80, ofreciendo ventajas muy sustanciosas con respeto a los agregados. El principal problema de estos modelos es el requerimiento matemático de estadística y econometría necesario para su correcta utilización. Sin embargo sus diferencias no son tan grandes. Esta recae en el tratamiento de la descripción del comportamiento durante el desarrollo del modelo.

El tipo de datos utilizados tiene su influencia estudiada, los tipos son datos transversales o datos longitudinales. En el caso 1, se trata de una instantánea del transporte en un intervalo concreto de tiempo, que no confiere ningún tipo de garantía de que este modelo pueda ser utilizado para predicciones sobre el futuro de transporte ante cambios realmente significativos. En el caso 2, los datos se toman desde diferentes etapas con el objetivo de evaluar la situación actual y su inercia con respecto a esta.

Las preferencias permiten modelar de diferentes formas las variables en función del conocimiento que se tenga del comportamiento de los usuarios. Las preferencias reveladas corresponden a aquellas obtenidas a través de encuestas recogiendo la información de los viajeros, revelando el comportamiento de los mismas al mostrar este sus decisiones y elecciones. Por otro lado, las preferencias declaras tratan de representar el comportamiento de los viajeros en una situación hipotética. A diferencia de las reveladas, las respuestas del encuestado responderían a situaciones como la introducción de una nueva opción de transporte en su zona o una mejora en la calidad del servicio.

Para finalizar, se expone el que se denomina *Classic Transport Model* que se trata de una estructura surgida en los años 60 después de mucha experimentación y pruebas de como se debe desarrollar un modelo de transporte. Resulta curioso que aunque se han desarrollado otras metodologías y técnicas este método sigue vigente y casi inalterado con respecto a su concepción inicial. Este modelo se presenta en la figura 2.2.

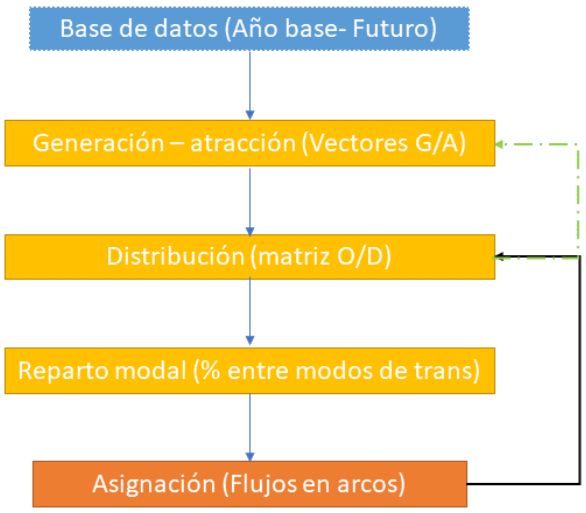

Figura 2.2 Diagrama del modelo de transporte.

La estructura de este modelo para seguir consecutivamente las etapas de las que se compone necesita de diversos y distintos datos para un correcto modelado del comportamiento de la zona en cuestión. Se tienen al comienzo de esta asignación la zonificación del territorio a analizar, los datos recogidos dependiendo del año a tener en cuenta, los datos de actividad económica en la ciudad incluyendo la actividad de empleo, los comercios, zonas educacionales y de recreo. Con esta base de datos se estima un modelo con el número de viajes generados y atraídos en cada área de estudio. Esta etapa se denomina *Trip Generation*. La distribución de estos viajes a zonas particulares produce una matriz origen destino distribuyendo los viajes en el espacio disponible. A esta fase se la denomina *Distribution*. El paso siguiente, conocido como *Modal Split*, requiere la modelización del proceso de toma de decisiones de los viajeros entre las diferentes modalidades de transporte.

Tabla 2.1 Diferentes clasificaciones asociadas con el problema de asignación.

Evolución de la demanda	Comportamiento de los usuarios		Influencia en las funciones de demora	
	Minimización de costes	Percepción de costes	Flujo - Distancia	Flujo Entre Arcos
Estático	Sistema óptimo	Determinista	Sin congestión	Simétrico
Dinámico	Equilibrio de usuario	Estocástico	Con congestión	Asimétrico

Por último, la asignación de los viajes o rutas en cada uno de los modos disponibles por las redes asociadas privadas o públicas. En el caso de que haya alternativas, se reparte el flujo entre ambas. Esta etapa es conocida como *Assignment*

El problema de asignación siendo el objetivo de este proyecto, puede clasificarse según las consideraciones expresadas en la tabla 2.1. Referente a la evolución de la demanda se refiere a si es cambiante o no, en otros términos, si se puede actualizar. Con respecto al comportamiento de los usuarios y como hacer un modelo para ello, se puede optar por minimizar los costes ,utilizando el sistema óptimo de *Wardrop* o el equilibrio de usuario; junto a percepción de costes, con un enfoque mas estadístico (estocástico) o con los costes completamente definidos y conocidos (deterministas).

La consecución de estas 4 etapas completa, usando cada etapa como submodelo, el modelo de transporte clásico de 4 etapas. Aunque de esta forma no es como se toman las decisiones en una atmósfera de trabajo real, es decir no siguiendo estrictamente esta secuencia, desde una vista mas contemporánea considera que la posición de cada submodelo depende de su función y utilidad asumidas para controlar o gobernar las decisiones de transporte. Además, el modelo de 4 etapas se percibe como un análisis de un amplio rango de respuestas para un limitado número de respuestas de viajeros.

Sin embargo, a pesar de los defectos que un modelo como el explicado previamente pueda tener, este modelo es tomado como un punto de referencia y como contraste para métodos alternativos lo cual describe su importancia. Aunque se debe recalcar que no es la única forma de realizar el modelo siguiendo las etapas en ese orden. Otros autores intercambiaron las etapas con resultados un tanto estridentes. Cambiando la importancia de variables, y relacionado de forma muy diferente los resultados extraídos del modelo.

Tras la calibración del mismo, se debe aplicar o tener en cuenta durante la planificación para distintos cometidos. La dificultad de construir escenarios reales para utilizar el modelo es alta debido a la alta posibilidad de caer en una trampa de construir escenarios futuros que no son viables desde la perspectiva económica.

Este proyecto se encuentra enmarcado la etapa de *Asignación*, y esta se debe hacer para el viaje público como el privado en el caso del modelo de 4 etapas. Se deben obtener los volúmenes en los arcos de la red de transporte dados:

- La matriz de origen destino o de viajes.
- Red de transporte donde es muy importante la verificación de la conectividad de cada par O-D comprobando que exista al menos una tura disponible. De lo contrario el programa arrojará la imposibilidad de redistribuir el flujo de viajeros entre ambos puntos, manteniéndose en su estado inicial sin cambio alguno [1].
- Las funciones de comportamiento de los arcos.

verificando todo esto las condiciones de equilibrio. Pero ¿Cómo se alcanzan las condiciones de equilibrio? La definición del transporte público se debe hacer en los mismos términos para el privado aunque las consideraciones son bastante diferentes. Por ejemplo: considerar el reparto de transporte público requiere de unas rutas, capacidades, frecuencias, tasas que difieren en gran medida de lo que se entiende por transporte privado.

2.2 Sistema de transporte en equilibrio

Para explicar un sistema de transporte en equilibrio se explicará de forma gradual haciendo al lector entender el concepto de forma progresiva. El equilibrio se puede tener en cuenta de formas muy diversas. La más simplista es en la que los viajeros buscan en una red de transporte desde una matriz de viajes fija la minimización de los costes de viaje. A esto le sigue una búsqueda de rutas alternativas buscando las opciones mas plausibles, llegando a un patrón después de un ensayo error muy sacrificado para el usuario. En este punto, se habría alcanzado el *equilibrio* puesto que no hay mejores posibilidades para escoger y todas las opciones se encuentran agotadas. Pero este no sería el equilibrio en términos generales de la red. Sería el de la red de

carreteras de transporte. Todavía admite mejoras como la inclusión de transbordos y tramos a pie que podría reducir el tiempo gastado andando. Adicionalmente, el viajero podría considera más parámetros como evitar la masificación en el vehículo, reducir la espera y los tiempos andando y en vehículo.

Cuando se introduce la congestión en el tráfico, algunas de las rutas que elegirían los usuarios en la situación anterior serían evitadas debido al atasco. Este tipo de equilibrio es conocido como *equilibrio de la red multimodo*.

Considerando la perspectiva de que esos parámetros afectan a los flujos de pasajeros, sus elecciones y destinos, se puede entender que desde las situaciones de equilibrio anterior los resultados cambian, aunque se tenga una matriz OD fija. Esto produce cambios en los niveles de servicio teniendo que calcular nuevas estimaciones de los parámetros es decir recalculando el equilibrio del sistema. El concepto que se le quiere explicar al lector no es otro que el de *sistema en equilibrio* en oposición a de red en equilibrio.

Los efectos de la congestión y el tráfico en el sistema urbano se aproxima a un sistema en equilibrio. Se trata de un sistema por lo explicado previamente, difiere ligeramente de la situación de equilibrio a través de cambios locales, como por ejemplo una ampliación de una avenida. Esto modifica los accesos, los vehículos que no la usaban ahora la utilizan, descongestionando otras zonas de transporte.

Tras esta situación de transición, puesto que el tráfico se adaptará a la nueva situación, se llega a un nuevo equilibrio, con tráfico recurrente y estable cada día. Se llega entonces al equilibrio *cuando no haya ninguna fuerza que tienda a llevar al sistema a otro estado*. Se puede diferenciar comprobando si alguno de los usuarios tiene la necesidad de cambiar de ruta, si el sistema estuviera en equilibrio no habría ninguna necesidad de cambiarla.

2.3 Equilibrio de *Wardrop* y paradojas extraídas del mismo

Knight en 1924 es el primero en definir el equilibrio en una red de tráfico de la forma explicada en la sección anterior [24]. Si se supone que entre 2 nodos hay 2 enlaces que los unen:

- Uno con capacidad para acoger cualquier volumen de tráfico pero pobremente pavimentado
- Otra con capacidad limitada pero exquisitamente pavimentado

Knight concluye que si un número de camiones tuviera libertad para escoger cualquiera de ellas, se distribuirían de tal forma que el coste por unidad transportada o el beneficio por unidad invertida sería el mismo para cada camión en cualquiera de las rutas. Mientras mas camiones usen el segundo enlace, se produce un incremento de la congestión que lleva a este sistema a un equilibrio de transporte o el beneficio obtenido en cualquiera de las vías en el mismo.

Evidentemente, para obtener este comportamiento la asunciones son:

- Cada viajero tiene información completa y precisa de todos los caminos disponibles y sus particularidades.
- El patrón de flujos sobre la red es estable sobre el tiempo de tal modo que la experiencia pasada por los viajeros resulta válida para su planificación viajera.

Esto define al modelo como determinista frente al modelo estocástico donde se asume que la información percibida por el usuario es incompleta. Además, queda definido como modelo estático dada la estabilidad de los patrones. Si estos cambiasen se definiría como dinámico.

Wardrop llega a unos principios muy importantes de para encontrar o definir el equilibrio en el artículo [25]. Estos únicamente coincidirán en el caso de que no exista congestión de ninguna clase. Las condiciones para el mismo:

- Equilibrio del usuario: no se puede mejorar el tiempo de viaje o el costo generalizado de un usuario cambiando de vía unilateralmente. La cual supone que ningún conductor sería capaz de reducir los tiempos seleccionando una nueva ruta.
- Equilibrio del sistema: en este equilibrio el tiempo de viaje es mínimo. Se tiene a un usuario comportándose de forma cooperativamente eligiendo su ruta de forma que asegura un uso más eficiente de todo el sistema.

Matemáticamente expresado y desde una perspectiva de la programación, la situación descrita por *Wardrop* (aunque fue *Beckman et al.* quien propuso un riguroso marco para expresarlo como programa matemático

[26]) puede ser expresada como la ecuación 2.2 sujeta a 2.3 donde se minimiza la función objetivo restringida a los límites impuestos por las propiedades de los flujos. En otras palabras, la función objetivo es la suma de las áreas bajo el coste del flujo para todos los *links* de la red.

$$Minimizar \qquad Z(T_{ijr}) = \sum_{a} \int_{0}^{V_a} C_a(\upsilon)\partial\upsilon \tag{2.2}$$

sujeta a:

$$\sum_{r} T_{ijr} = T_{ij}; \qquad T_{ijr} \geq 0 \tag{2.3}$$

donde T_{ijr} es el flujo en una ruta concreta y en la integral $C_a(\upsilon)$ es el coste por unidad de área en el enlace correspondiente.

Estos principios son los que rigen la conducta de los conductores en sus elecciones de ruta. Si estas condiciones no se cumplen la condición de equilibrio no ha sido alcanzada pudiendo mejorarse los tiempos. Ante este problema, la solución que surgió después de verificar que resolver el problema mediante álgebra no era una opción, el esquema iterativo de resolución cobró cierto atractivo tratando de dar soluciones que se aproximan a la situación descrita por *Wardrop*. En este tipo de esquemas se encuadra el algoritmo programado.

Las ideas básicas de primer y segundo principio quedan ilustradas en la *paradoja de Braess* [27] que concierne al transporte privado por las condiciones en las que se da. Aunque no se trata de un paradoja en sentido estricto, demuestra bajo ciertas condiciones (flujo de vehículos alto) añadir capacidad a una red de carreteras hace que cuando los conductores busquen minimizar sus costes propios, resulta en un empeoramiento de las condiciones de todos. Por ejemplo: la ciudad de Stuttgart buscaba en los años 60 mejorar el tráfico del centro con la apertura de una calle nueva para aliviar el transporte. Sin embargo, el efecto conseguido fue el contrario, la congestión aumentó conllevando un empeoramiento del tráfico. Sorprendentemente cuando decidieron cerrarla el tráfico volvió a mejorar.

Como ejemplo ilustrativo se utiliza la figura 2.3. Se tiene una demanda del *nodo 1* al nodo *nodo 4*. La situación inicial (sin el arco discontinuo que aparece en la figura citada) alcanza el equlibrio con el flujo con valor 3, con un coste para el viajero de 83. Siendo el coste total de la red 83 multiplicado por 6. La situación final, con el arco discontinuo en funcionamiento, representa la respuesta de la red anterior al *link 2-3*. Aquí el equilibrio se alcanza con flujo por cada uno de los caminos de 2 pero con un coste por viajero de 92. Siendo el coste de la red 92 multiplicado por 6.

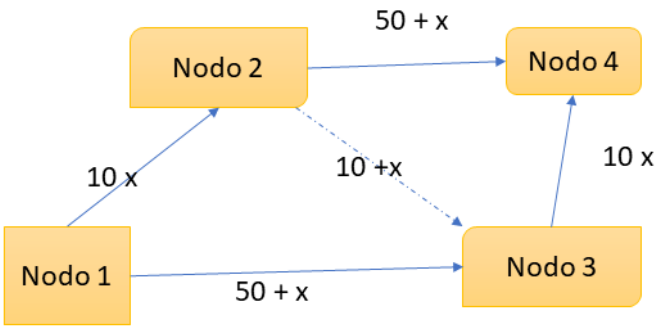

Figura 2.3 Diagrama de la paradoja de *Braess*.

Por otro lado, una paradoja que se extrae del transporte público conocida como *Downs-Thomson ó Pigou-Knight-downs* establce que bajo unas condiciones concretas (una gran demanda de transporte que podría ser hora punta en un corredor de conmuters utilizando 2 modos de transporte que no comparten red [28]. *Mogridge* se encargó de documentarlo de forma muy detallada en el documento citado y su libro [29]. Uno de los ejemplos utilizados en ese libro es el ejemplo del *Public Central* de Londres donde desde 2001 en la hora punta de transporte, el 85 % de los viajeros de esa mañana utilizaban el transporte público y el 11 % solo utilizaba transporte privado. Si la decisión de invertir se centra en carreteras, la mayor capacidad atraerá

a usuarios del transporte público al uso del coche, que reducirá la calidad de su servicio por la disminución de la cantidad de pasajeros, y en particular, la densidad de las rutas y frecuencias. Esto volverá a repetirse, induciendo una mayor congestión con la contrapartida de un sistema de transporte empeorado generando un círculo vicioso. Esta cualidad destructiva consigo mismo de la política de capacidad de las vías urbanas es lo que describe la paradoja. Queda esquematizado en la figura 2.4.

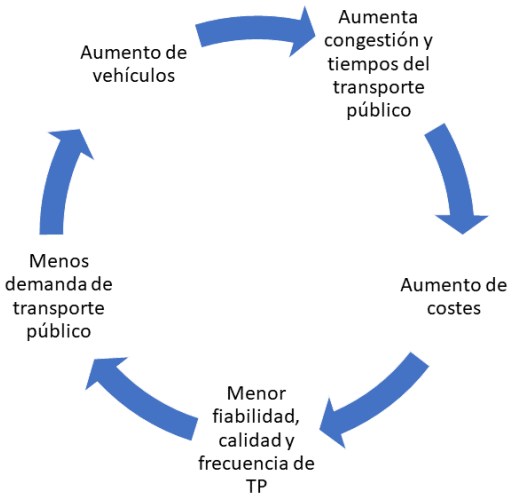

Figura 2.4 Diagrama del círculo vicioso de *Downs-Thomson*.

La paradoja explica que las mejoras en las carreteras que utilizan el transporte no reducen la congestión del tráfico. De facto, las mejoras pueden llegar a empeorarlo si esta afectando al transporte público haciéndolo menos conveniente o si cambiando el objetivo de las inversiones reduciendo la cantidad sobre el mismo.

2.4 Transporte privado y Transporte público

Los conceptos de oferta y demanda son necesarios para completar la visión del lector sobre la asignación del sistema público. Se debe recordar que una red de transporte siempre soportará a una pública y otra privada pero se diferenciarán en la información que se proporciona de cada una. Para el caso del público, se necesita saber los servicios que ofrece especificados como rutas, capacidades, frecuencias, tasas, tiempos, y aunque en la realidad sea algo escaso, la calidad, fiabilidad y regularidad del servicio. De hecho, es esta oferta de red de transporte público (que contiene a la *Asignación*) la que, inmersa en un esquema iterativo, se utiliza para ajustarse a la demanda (que viene definida por las 3 primeras etapas de *modelo clásico*). Mediante iteraciones, se consigue adaptar la oferta a la demanda que se ha estimado dependiendo de los datos de entrada de los que conste el modelo previamente en el caso de creación de nuevas líneas. Lo habitual es a la inversa, la demanda se adapta a la oferta. Puesto que son los usuarios los que optimizan sus viajes y elecciones frente a lo que ofrece el transporte público.

En este proyecto, se hará hincapié en el transporte público por motivos de la extensión del mismo y la zona del campo del diseño de trasporte que abarca.

Otra gran diferencia entre la red de privados y públicos, es que el camino de un punto a otro está siempre detallado como subenlaces entre las paradas disponibles (en el caso público). El concepto de la capacidad de un enlace está relacionado con la capacidad de cada autobús y su correspondiente frecuencia. El tiempo de viaje contiene mas elementos como por ejemplo: tiempo a pie, tiempo esperando el bus, caminando en contra con lo encontrado en el transporte privado. Las red de transporte también es diferente, se puede incluir bus, tren, metro en el público por lo que su naturaleza es muy diferente de la del privado produciendo unas redes de transporte mucho mas complejas. Un ejemplo de la complejidad se puede apreciar en el plano actual de transporte público de Lisboa 2.6.

Tratar los datos de pasajeros con la posibilidad de desplazamiento a pie es la principal diferencia encontrada con el privado. Los transbordos, la comunicación a pie entre zonas esta disponible ante el retraso de un vehículo. Lo cual expresa la necesidad de diseñar y obtener una red de transporte lo suficientemente densa para simular el comportamiento de un peatón. Se ha conseguido este efecto utilizando *Google Maps*.

Frente al coste monetario obtenido de la gasolina gastada en el transporte privado, los usuarios del sistema público no perciben tanto ese gasto de forma tan directa porque solo compran un billete al conductor del vehículo. Con el surgimiento de las nuevas tecnologías, el pago mediante tarjetas de bus o con el teléfono móvil, la incorporación de un sistema de tarifas mas diverso es posible. Desde la modalidad de zona, billete de tarifa plana entre otros son los ejemplos de tickets que se pueden encontrar en los sistemas actuales. Este aumento de la diversidad en tarifas incrementa la complejidad del problema de asignación porque la dependencia directa de la longitud cubierta pasa a depender de su origen destino.

Figura 2.5 Diagrama del transporte urbano de la ciudad de Lisboa.

Respecto de la variable costes C_{ij} que mencionada en varias ocasiones durante el proyecto, se define que costes se consideran en un viaje en la ecuación 2.4. Donde t_{ij}^{v} es el tiempo en vehículo desde i hasta j, t_{ij}^{w} es el tiempo a pie desde y hacia las estaciones de transporte público, t_{ij}^{t} es el tiempo de espera en paradas, t_{ij}^{n} es el tiempo de transbordo, δ^{n} es la resistencia al transbordo, F_{ij} son las tarifas desde i hasta j, por último, de a_1 a a_5 son los coeficientes asociados a los elementos del coste generalizado.

$$C_{ij} = a_1 t_{ij}^{v} + a_2 t_{ij}^{w} + a_3 t_{ij}^{t} + a_4 t_{ij}^{n} + a_1 \delta^{n} + a_5 F_{ij} \tag{2.4}$$

En el caso de este proyecto hay unos cambios en 2.4 debido al alcance y cometido del proyecto:

- Los costes asociados a las tarifas no se tienen en cuenta por el alcance que tiene el proyecto y el tiempo de duración del mismo.
- La penalización de transbordo se ha realizado de otra forma. Mientras que todas las constantes de a_1 a a_3 son 1, los términos 4 y 5 de la ecuación son eliminados imponiendo $a_4 = 1.2$ para simular con mayor realismo el comportamiento de los peatones expresados en el flujo de la red

A continuación el modelo para simular el transporte público con el mayor realismos se enfrenta al problema de decidir entre varias lineas que permiten ir desde un punto a otro del mapa. Se definirá el problema de *elección de rutas* que es clave en la *etapa de asignación* del modelo de 4 etapas. Una linea de transporte son vehículos haciendo un recorrido entre 2 puntos dentro de una red de transporte. Suelen tener características parecidas de tamaño, capacidad de viajeros, velocidad, entre otros. Paran para recoger o dejar viajeros, circulan con una cierta frecuencia regida por un horario de transporte. Una porción de linea es una parte del transporte público entre 2 nodos, no necesariamente consecutivos, en una red de transporte público.

Por lo tanto, una ruta de transporte público es un camino que cualquiera de los usuarios puede seguir en la red para viajar entre 2 nodos. Una parte de esta ruta es una porción de ruta, cada una contiene un conjunto de

lineas atractivas o comunes. Sabiendo esto los modelos de asignación de transporte público pueden dividirse en:

- *Todo o nada* que se utilizan para lineas de larga distancia y distribuidas de forma muy lejana.
- Los modelos *multipath* que establecen el flujo de pasajeros entre los posibles caminos dependiendo de las frecuencias que posean las lineas.
- Modelos de asignación basados en el equilibrio con elementos estocásticos o no. Se utilizan para estudiar la congestión en sistemas de transporte público.

Este proyecto se engloba en la segunda opción, un modelo *multipath*, flexible, que considera el cambio del tiempo de espera y la oportunidad de elegir servicios de tiempo de viaje bajos, pero con baja frecuencia. Este problema tiene una resolución en forma de algoritmo que cobra la forma:

1. Se fija un nodo de destino i.
2. Coger el primer transporte que llegue de el conjunto de las lineas atractivas en i.
3. Bajarse en un nodo predeterminado.
4. Si este nodo no es el de destino, fijar i como el nodo actual y volver al paso 2. En caso contrario, se ha llegado al destino.

Es importante mencionar que aunque el nodo de destino esta fijado no es parte de la estrategia. Una estrategia es un conjunto de normas que permite a un viajero llegar a su destino desde cualquier nodo de la red.

La resolución de este problema mediante programación (obtención de la estrategia óptima), requiere de solucionar 2.5 sujeta a 2.6, 2.7 y 2.8. Donde S_{ij} conjunto de secciones de lineas que conectan los nodos i a j directamente, L_j^+ conjunto de secciones de lineas desde el nodo j (si el superíndice es $-$ las lineas son entrantes), v_s el flujo en la sección de linea s, t_s es el tiempo de viaje en un vehículo en una sección de linea s, f_s es la frecuencia asociada a la sección de linea s, g_j es el número de viajes hacia el nodo j, y por último, V_{ij} es el flujo total en la sección de ruta jk.

$$Minimizar \qquad \sum_s v_s t_s + \sum_{jk} w_{jk} \tag{2.5}$$

$$\sum_{s \in L_j^+} v_s + g_j = \sum_{s \in L_j^-} V_s \tag{2.6}$$

$$v_s = \frac{X_s f_s V_{jk}}{\sum_{s \in S_{ij}} f_s X_s} = X_s f_s w_{jk} \tag{2.7}$$

$$w_{jk} = \frac{V_{jk}}{\sum_{s \in S_{jk}} f_s X_s} \tag{2.8}$$

La ecuación 2.8 define el tiempo total de un viajero para ir de j a k. 2.5 es la función objetivo, su primer término es el tiempo total en vehículo por ruta de todos los usuarios del sistema de trnasporte. El segundo es el tiempo de espera total, con lo que la función objetivo es el tiempo total en llegar desde los orígenes a los respectivos destinos. La función es lineal en v_s y w_{jk} pero el problema se deriva de las restricciones no lineales 2.6, 2.7 y 2.8. En el artículo [1] *Spiess y Florian* demuestran que se pueden relajar estas restricciones mediante la ecuación 2.9.

$$v_s \leq f_s w_{jk} \tag{2.9}$$

Introduciendo las restricciones 2.7 en la función objetivo se obtiene la función a minimizar 2.10 sujeta a 2.6.

$$Minimizar \qquad \sum_{jk} \frac{V_{jk}(\sum_s t_s X_s f_s + 1)}{\sum_{S \in S_{ij}} f_s X_s} \tag{2.10}$$

De este problema los modelos mas relevantes son:

- *Spiess y Florian*[1] que resuelven una versión lineal del problema quedando implementado en *EMME/2*, siendo una contribución muy novedosa al campo. Si no hubiera problemas de capacidad o congestión el problema no dependería de los flujos de transporte.

- El algoritmo de *De Cea-Ferández* [30] implementado en *ESTRAUS* resuelve el problema no lineal al completo siendo 2.5 veces mas rápido que el de *Spiess y Florian*.

Se ha implementado en este proyecto el algoritmo de *Spiess et al.* [1] para la simulación de las rutas óptimas de Sevilla con las lineas consideradas puesto que es la propuesta del departamento de transportes de la Universidad de Sevilla. Los siguientes pasos en esta línea serán la mejora del modelo de asignación considerando restricciones de capacidad en los vehículos.

Cuando las estrategias óptimas han sido resueltas se procede al cálculo de asignación mediante 2 etapas(para los casos previamente vistos este paso es muy similar):

1. Se construye con las rutas óptimas (únicamente las que minimizan el transporte) una nueva red de transporte sobre la que se distribuye el flujo de pasajeros que circularía por los caminos óptimos

2. Los flujos de secciones de ruta pueden ser descompuestos en las secciones de linea mediante 2.11.

$$v_s = \frac{f_s v_r}{f_r} \tag{2.11}$$

Finalmente, la red de transporte público sufre de las mismas debilidades que la las redes de carreteras. Por dos motivos: el primero, la capacidad limitada de vehículos puede hacer que algunos viajeros no implementen sus estrategias óptimas; y segundo, la interacción entre coches privados y vehículos públicos compartiendo la misma red de carreteras incrementa el tráfico de forma que puede afectar a los tiempos de ambos.

Respecto de las limitaciones de los modelos clásicos, se han reducido las siguientes:

- El efecto *rotonda* por el que se considera una rotonda modelada de una forma que en la realidad no existe (ver figura 10.12 [2]). Al obtener los tiempos desde *Google Maps* no hay reducciones de tiempo ni cambios con respecto a los reales, únicamente simplificaciones visuales.
- Las variaciones de un día a otro de una fecha cualquiera quedan solucionadas mediante la extracción de datos de *Google Maps* puesto que se pueden obtener en tiempo real desde una fuente de datos casi a tiempo de usuario.
- Limitación de tener que actualizar los resultados incluidos en el modelo para un cálculo con ellos. No se podría implementar un tiempo real en sentido estricto por la cantidad de datos y el tiempo de cálculo.
- Errores de entrada de datos. Algunos datos (algunas minucias no han sido filtradas) no han podido ser corregidos por motivos ajenos al autor del proyecto.
- La red de transporte ha sido comprobado 3 veces en búsqueda de errores tratando de eliminar el mayor número de errores posibles.

2.5 Tiempo de espera

Respecto del 2.8 tiempo de espera, la solución de *Spiess y Florian* sobre como considerarlo se explica a continuación.

El tiempo medio de espera posee 2 componentes aleatorias fuertes: *la aleatoriedad de llegada del pasajero a la parada* y *la aleatoriedad del tiempo entre 2 llegadas*. Para considerar este tiempo de forma adecuada, se le considera al entre llegadas como una variable aleatoria, puesto que la congestión vuelve aleatoria el esquema oficial de la linea de transporte. Habrá una función de densidad conjunta de las variables y_i (longitud o duración de los intervalos entre llegadas) como se expresa en la ecuación 2.12 y usando la figura 2.6 como referencia.

$$\beta_{Y_1,Y_2,Y_3,\ldots Y_N}(y_1,y_2,y_3\ldots y_n) \tag{2.12}$$

No se hace ninguna hipótesis sobre β de distribución de todas las variables. Tomando este como punto de partida, se supondrá que las funciones de densidad marginales de cada variable Y_i son todas idénticas. Aún

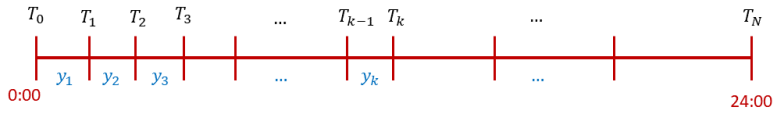

Figura 2.6 Esquema con la distribución de tiempos durante un día.

no se ha impuesto la independencia, únicamente la identidad de su distribución (eq 2.13) para expresarlo como caso denominado *Renewal process*.

$$\beta_{y_1}(x) = \beta_{y_2}(x) = \ldots = \beta_{y_k}(x) = \ldots = \beta_{y_N}(x) = \beta(x) \tag{2.13}$$

Se supone que un pasajero potencial llega en un momento aleatorio (*Random Incidence*) y se desea conocer la β de densidad de la variable tiempo de espera hasta que llegue el siguiente autobús. Se explicará a través del caso discreto, para generalizar obteniendo lo que atañe, el caso continuo.

¿Cuál es la probabilidad de que el tiempo de espera sea un valor concreto w?

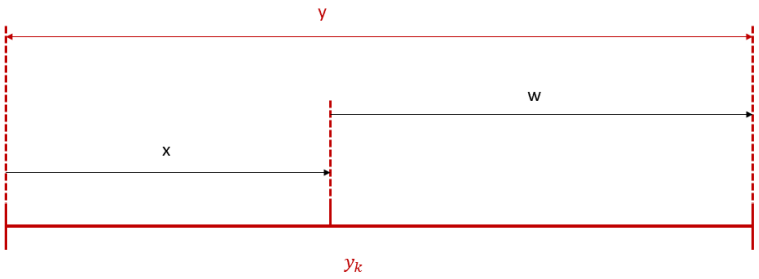

Figura 2.7 Esquema probabilidad del tiempo de espera como valor concreto.

Con ayuda de la figura 2.7, para obtener la probabilidad $P(W = w)$, el usuario debe llegar en un intervalo y de duración mayor que ese tiempo de espera w, y además, llegar en el primer tramo definido por $x \leq y - w$.

Como se conocer la β de distribución de los intervalos entre llegadas, se conoce la probabilidad de llegar en un intervalo de una longitud concreta y. Pensando en el caso discreto no es directo $\beta(y)$, ya que por ejemplo en el mismo día han podido haber varios intervalos de esa longitud y, y cualuiera de ellos es válido como se expresa en la figura 2.8.

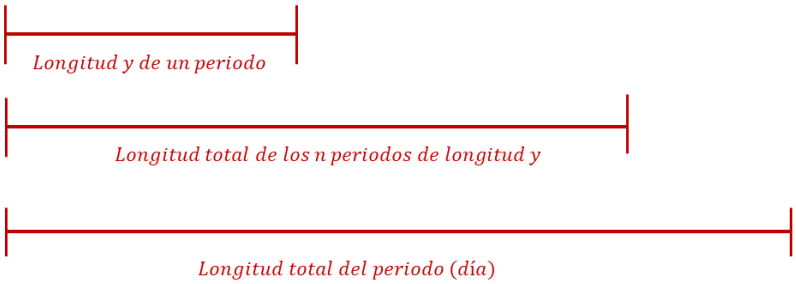

Figura 2.8 Varias posibilidades de intervalos y.

La probabilidad de que se llegue en un intervalo de longitud y puede expresarse de la forma esgrimida en la ecuación 2.14. En su segunda equivalencia $\frac{N}{L}$ representa la longitud total frente al número de periodos,

que puede deducirse como la longitud media $\bar{y}$. Por lo tanto $K = \frac{1}{\bar{y}}$. En la tercera equivalencia, $\frac{n}{N}$ representa la frecuencua de aparición de esos intervalos.

$$P = \frac{yn}{L} = \frac{N}{L} y \frac{n}{N} = K\, y\, P(Y = y) \tag{2.14}$$

2.14 expresada en el caso continuo queda 2.15.

$$h(y)\, \partial y = K\, y\, \beta(y) \tag{2.15}$$

Se puede apreciar que K, para que $h(y)$ una función β de densidad y su integral $\int_0^\infty K = 1$, tiene que ser, de nuevo como el caso discreto, la inversa de $\bar{y} = E(y) = \int_0^\infty y\, \beta(y)\, \partial y$.

Es muy importante resaltar que las β dan las frecuencias de aparición de los intervalos, no la probabilidad de que en un instante concreto estemos en un intervalo de una longitud determinada.

Resta comprobar la probabilidad de caer en el primer tramo del intervalo condicionada a que la llegada se produce en un intervalo de longitud y. Como se supone que puede llegar en cualquier momento, no hay picos de probabilidad. Por lo tanto, todos los instantes de tiempo son equiprobables. En otras palabras, se tiene una *distribución uniforme* de probabilidad.

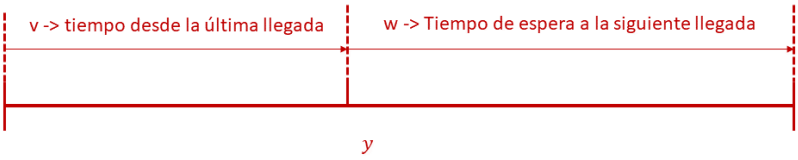

Figura 2.9 Esquema de llegada para el calculo de β condicionada.

En el caso de la función conjunta de probabilidad $\beta_{v,y}$ se utiliza el esquema presentado en la figura 2.9 cuya ecuación corresponde a la mostrada en 2.16. Se recuerda también que en la figura 2.9 el intervalo queda definido $0 \leq w \leq y$, y además, $\beta_{w/y} = \frac{1}{y}$.

$$\beta_{V,Y}(w,y) = \beta_{W/Y}(w/y)\, h(y) = \frac{1}{y} \frac{y\, \beta(y)}{\int_0^\infty y\, \beta(y)\, \partial y} = \frac{\beta(y)}{\int_0^\infty y\, \beta(y)\, \partial y} \tag{2.16}$$

Hasta 2.16 se ha obtenido la función de densidad del tiempo de espera para una línea concreta. A continuación, se calculará la esperanza de una variable aleatoria de la que se conoce su función de densidad F pero no su función de densidad f.

Realmente, el interés se centra en la que depende de la variable tiempo espera w. Por lo tanto integrando 2.16, se obtiene la ecuación 2.17.

$$g(w) = \frac{\int_w^\infty \beta(y) \partial y}{\int_0^\infty y\, \beta(y) \partial y} = \frac{1 - F(w)}{E(Y)} \tag{2.17}$$

2.6 Capacidad

La capacidad de una red de transporte es uno de los factores principales con los que se obtendría la velocidad del sistema que opera junto a la gestión del modelo y el volumen. Como definición formal es la cantidad de viajeros que puede gestionar y albergar un sistema de transporte público en los elementos que lo componen. El fenómeno de congestión aparece cuando el número de usuarios se aproxima a la capacidad de diseño del sistema. Por lo tanto, se puede apreciar la importancia de este concepto en un modelo puesto que define los límites del sistema bajo unas condiciones determinadas. Puede entenderse como la máxima intensidad de una vía sin llegar al colapso. Esta depende principalmente de:

- La gestión se debe encargar de un correcto uso, redistribución y una eficiencia cuando se utilice el transporte urbano.

- La inversión se define por la aportación económica y facilidades que el Gobierno y otras corporaciones de índole privada o pública

En el transporte público, representa uno de los factores más relevantes junto a la frecuencia del mismo en cuanto a la oferta. Se puede entender mejor como concepto exportable. Se utiliza también para definir el número de pasajeros en vehículo cada hora que puede albergar un enlace. También define los vehículos en una vía por hora. Además, es uno de los factores que influye a los costes generalizados estando relacionado por la definición que se hace del enlance en el modelo de Spiess.

3 Peticiones a Google Maps

Passion is destructive; if it does not destroy, it dies.

William Somerset Maugham

Dentro de este capítulo se explica como realizar peticiones a *Google Maps* mediante sus *APIs* [31, 8]. Se definirán los tipos, la información que proporciona cada petición, como se han analizado esa información y la forma de organizarla dependiendo de la consulta.

3.1 Que es una petición, como generarla y sus tipos utilizados

3.1.1 Definición de petición

Una petición es una consulta a una base de datos especificando los detalles definidos por el lenguaje de programación para concretar la información resultante que se quiere obtener [32]. Por ejemplo: en el caso de este proyecto las peticiones realizadas han sido para obtener coordenadas de sitios concretos en la ciudad de Sevilla como pueden ser las paradas de autobús por las que tiene que pasar una línea. Se podría hacer copiando y pegando la información obtenida a un fichero, almacenándola de forma organizada pero dada la magnitud del problema no es recomendable y ni fiable, puesto que realizando *a mano* el proceso aumenta considerablemente la probabilidad de cometer un fallo humano, y con esto, la obtención de base de datos incompleta como resultado es más que factible. Todo esto, sin considerar que es un proceso enormemente lento. Por ejemplo, se adjunta una dirección web donde consultar un ejemplo de petición para *Google Maps* desde el modo *directions* [32]

Por lo tanto, la solución que elimina los problemas citados y acelera el proceso de una forma satisfactoria es programar las peticiones y aprender a analizar la información que proporcionan las consultas.

En el presente caso, siendo un proyecto que trabaja con datos de transporte público de Sevilla capital [33], lo más lógico, novedoso y fidedigno es utilizar los datos obtenidos desde las aplicaciones que *Google* tiene disponibles para consultar la información que se necesite. Además, *Google* obtiene los datos de forma directa de la empresa encargada de suministrar y gestionar el transporte público de Sevilla, cuyo nombre es *Tussam*. Por lo que extraer los datos de esta forma, permite elaborar una base de datos completa, con información de tiempos casi real (ya que no será un problema resuelto de tiempo real en lo que a las consultas en el mismo momento instante al se refiere) que proveerá con lo necesario para realizar la asignación de transporte, resolviendo así el problema.

3.1.2 Como construir y realizar una petición usando APIs de Google

Para cumplir este cometido de forma correcta se tienen los siguientes pasos:

1. Acudir a la información proporcionada por *Google* en su web para conocer como se formula la petición, la información que va a arrojar [31, 32]
2. Crear la petición utilizando *Mathematica* junto con los detalles que se hayan escogido para definir la petición. A continuación en 3.1, un extracto del código utilizado en el proyecto se utiliza para mostrar

cómo es generar una petición. Se observa que la petición es un enlace URL con unos parámetros especiales como es el caso de 3.1. Se puede apreciar que: el tipo de petición según su seguridad que puede ser *http* ó *https*, *Directions* es el tipo de API utilizada en la consulta, el formato en que se quiere recibir los resultados puede ser *XML* ó *JSON* (siendo escogido el segundo en este caso), el atributo *key* que permite realizar peticiones utilizando todos los detalles proporcionados por *Google* en su guía [8] y los parámetros restantes se pueden consultar en la referencia específica [32]. Un ejemplo de petición genérica queda representado en 3.2 junto a la referencia [34]

Código 3.1 Ejemplo de petición.

```
// Programming a Google Maps Query
Needs[GeneralUtilities‘]
peticionTransitGooglelcortas[l_]:=https://maps.googleapis.com/mapsapi/
    directions/json?origin=<> ToString[First@First@l]<>,<> ToString[
    Last@First@l]<>&destination=<>ToString[First@Last@l]<>,
<>ToString[Last@Last@l]<>&key=YOUR_KEY_HERE<>&mode=transit<>
&alternatives=true<>&transit_mode=bus<>
&transit_routing_preference=less_walking
```

Código 3.2 Petición genérica utilizada en Google.

```
// Generic Query
peticionGenerica[l_]:=https://maps.googleapis.com/maps/api/service/output?
    parameters
```

3. Utilizando el siguiente extracto de código se ejecuta el código obteniendo directamente la información de la petición en la variable **Query**.De esta forma habríamos obtenido la información desde Google. En la primera linea de 3.3 inmediatamente superior se tendrían las coordenadas exactas del inicio y final de la linea 1 de la red de transporte de Sevilla.

Código 3.3 Ejecutando una petición.

```
// Executing a Mathematica URL generated
l = {{37.411145, -5.983429},{37.3572018,-5.9804376}}
Query = URLExecute[peticionTransitGooglelcortas[l]];
```

Posteriormente se definirán los tipos de peticiones que se han realizado durante el desarrollo del proyecto. Es importante conocer que toda la información sobre peticiones queda recogida y como procesarla [34].

3.1.3 Clases de peticiones se han utilizado

Se dividen en 3 grupos principales:

- Peticiones tipo *Directions*. Se utilizan para obtener datos sobre el tipo de viaje que se quiere realizar desde un punto inicial a un punto final [21]. En concreto, *Maps* funciona resolviendo varios tipos de problemas como los citados en [35]. El directorio donde se especifican los problemas junto en el citado en la previa frase, tiene su variante en *Github* donde la mayoría de las funciones utilizadas para resolver el problema están expuestas y accesibles para todo el público [36]. De esta forma, este tipo de peticiones proporcionan una parte de la información necesaria para construir la red de transporte para resolver el problema de asignación.
- Peticiones tipo *Google Places*. En concreto, las denominadas *Nearbysearch* [20]. Se utilizaron para para resolver junto a una rutina de filtrado los intrigantes y problemas del modo *walking* para los transbordos, accesos a pie y entre otro tipo de enlaces entre los puntos de la red de transporte. Por medio de sus centroides con un radio de 750 metros, se detectaron e incluyeron todas paradas bajo ese área circular proporcionando las paradas que poseen un modo pie.

- Peticiones tipo *Javascript*. Este tipo de peticiones se aprendió pero no ha sido utilizado como extractor de información, su función ha sido principalmente la de comprender como funcionan las peticiones, formularlas de un modo diferente, y además, durante un tiempo del proyecto se postulaba como una posible solución a los problemas de filtrado de datos y programación que se enfrentaron. Finalmente, es cultura general para el usuario. Toda esta sabiduría queda recogida en [12].

Obtención de la *key* de *Google* para peticiones muy detalladas

El proceso para obtener una clave de *Google* de una aplicación concreta requiere primero de encontrar cual es la API indicada para el cometido a alcanzar. Accediendo a [34] se puede encontrar la aplicación necesaria.

Seguidamente, obtener una clave es sencillo accediendo a [37]:

1. Ingresa a *Google API Console*
2. Crea o selecciona un proyecto
3. Haz click en Continue para habilitar la API
4. En la página Credentials, obtén una **clave de API** (y configura las restricciones para esta)
5. Para evitar robo de cuota acceder a [38]
6. Comprobar que no tenemos activados ningún ajuste de facturación para que *Google* no pueda aceptar infinitas peticiones en un día pero no cobre por ello. Para información mas detallada visitar [39] donde se explican las condiciones de obtener una clase junto a sus reglas de uso y límites.

3.2 Accediendo a los resultados, su estructura, su análisis y su filtrado

Para acceder a los resultados tras realizar una petición es necesario estudiar su estructura de datos (puesto que Google Maps define los campos que proporciona pero no como están esquematizados). Este sencillo análisis se hace utilizando *Mathematica* para analizar la estructura de los mismos para diseñar un programa que automatice el proceso de obtención de datos y su posterior guardado. Los campos de los resultados obtenidos junto con la respuesta en la codificación espera se puede consultar en [21].

Se sigue el siguiente proceso:

- Se estudia la organización de los datos obtenidos mediante las funciones de *Mathematica Cases Dimensions* y *Head* [14] de la forma expuesta en 3.4 obteniendo *geocoded waypoints, routes, status*, los cuales son los próximos a estudiar detenidamente.

Código 3.4 Trabajando con los datos al principio.

```
// How to extract the structure info
Cases[Output_info,Rule[x_,y_]->x]
```

- Se ejecuta *Output info* de la forma explicada en 3.5 para comprobar el contenido de la información. La información importante se encuentra en el campo *routes* por lo que el estudio hará hincapié en la misma.

Código 3.5 Estudiando la estructura de Output info.

```
// Structure command viewer
Output_info[[1]]
```

Con motivo de ser ilustrativo y mostrar un ejemplo, este campo tiene la apariencia mostrada en 3.6.

Código 3.6 Estructura de *routes*.

```
// Appearance of routes field
{warnings -> {Walking directions are in beta. Use caution \
```

```
\[Dash] This route may be missing sidewalks or pedestrian paths.},
 bounds -> {northeast -> {lat -> 37.4045, lng -> -5.98614},
   southwest -> {lat -> 37.3564, lng -> -6.00363}},
 copyrights -> Map data 2017 Google, Inst. Geogr. Nacional,
 overview_polyline -> {points ->
    kd_cF‘fpc@}‘@u@cUzF_LoD}OxIyW~MaSpKwYfOiK|IsJhKcU~KsM‘K{O~\
BgMqIwIgGmPwN{UeSCDTJdAVJi@JILALB@IJuABQ},
 legs -> {{departure_time -> {text -> 1:30pm,
     time_zone -> Europe/Madrid, value -> 1500636628},
    duration -> {text -> 30 mins, value -> 1777},
    arrival_time -> {text -> 2:00pm,
      time_zone -> Europe/Madrid, value -> 1500638405},
    end_location -> {lat -> 37.4037, lng -> -5.99426},
    distance -> {text -> 6.1 km, value -> 6146},
    end_address -> Calle Resolana, 39B, 41002 Sevilla, Spain,
    start_address ->
     Av. de la Reina Mercedes, 10, 41012 Sevilla, Spain,
    start_location -> {lat -> 37.3564, lng -> -5.98641},
    steps -> {{transit_details -> {departure_time -> {text ->
           1:31pm, time_zone -> Europe/Madrid,
          value -> 1500636660},
         departure_stop -> {location -> {lat -> 37.3564,
            lng -> -5.98641},
          name -> Reina Mercedes (Gta. Alc. Parias Merry)},
         arrival_time -> {text -> 1:58pm,
          time_zone -> Europe/Madrid, value -> 1500638280},
         arrival_stop -> {location -> {lat -> 37.4045,
            lng -> -5.99487},
          name -> Concejal Jiménez Becerril (Barqueta)},
         headsign -> Pino Montano,
         line -> {agencies -> {{name ->
              Transportes Urbanos de Sevilla S.A.M.,
             phone -> 011 34 955 47 90 00,
             url -> http://www.tussam.es/}},
          vehicle -> {icon ->
            //maps.gstatic.com/mapfiles/transit/iw2/6/bus2.png,
           name -> Bus, type -> BUS},
          name -> Bellavista-S. Jeronimo-Pino Montano,
          short_name -> 03}, num_stops -> 17},
       end_location -> {lat -> 37.4045, lng -> -5.99487},
       distance -> {text -> 6.0 km, value -> 5998},
       travel_mode -> TRANSIT,
       start_location -> {lat -> 37.3564, lng -> -5.98641},
       html_instructions -> Bus towards Pino Montano,
       duration -> {text -> 27 mins, value -> 1620},
       polyline -> {points ->
          kd_cF‘fpc@uMYgR[cUzF_LoD}OxIyW~MaSpKwYfOiK|IsJhKcU~KsM‘K{O~\
BgMqIwIgGmPwN{UeS}}, {end_location -> {lat -> 37.4037,
         lng -> -5.99426},
       distance -> {text -> 0.1 km, value -> 148},
       travel_mode -> WALKING,
       start_location -> {lat -> 37.4045, lng -> -5.99487},
       html_instructions ->
        Walk to Calle Resolana, 39B, 41002 Sevilla, Spain,
       duration -> {text -> 2 mins, value -> 113},
       polyline -> {points ->
          {phcF|zqc@CDFBLFPFr@NF]BKBCFEDAF?LB@IH}@@WBQ},
```

```
      steps -> {{end_location -> {lat -> 37.404,
          lng -> -5.99508},
         distance -> {text -> 54 m, value -> 54},
         travel_mode -> WALKING,
         start_location -> {lat -> 37.4045, lng -> -5.99487},
         html_instructions ->
          Head <b>southwest</b> on <b>Av. Concejal Alberto \
Jiménez-Becerril</b>,
         duration -> {text -> 1 min, value -> 39},
         polyline -> {points ->
            {phcF|zqc@CDFBLFPFr@N}}, {end_location -> {lat ->
            37.4038, lng -> -5.99483},
         distance -> {text -> 43 m, value -> 43},
         travel_mode -> WALKING,
         start_location -> {lat -> 37.404, lng -> -5.99508},
         html_instructions ->
          Turn <b>left</b> toward <b>Calle Resolana</b>,
         duration -> {text -> 1 min, value -> 37},
         maneuver -> turn-left,
         polyline -> {points ->
            cnhcFf|qc@F]BKBCFEDAF?LB}}, {end_location -> {lat \
-> 37.4037, lng -> -5.99426},
         distance -> {text -> 51 m, value -> 51},
         travel_mode -> WALKING,
         start_location -> {lat -> 37.4038, lng -> -5.99483},

         html_instructions ->
          Turn <b>left</b> onto <b>Calle Resolana</b><div \
style=\font-size:0.9em\>Destination will be on the right</div>,
         duration -> {text -> 1 min, value -> 37},
         maneuver -> turn-left,
         polyline -> {points -> olhcFtzqc@@IH}@@WBQ}}}}},
   traffic_speed_entry -> {}, via_waypoint -> {}}},
 summary -> , waypoint_order -> {}}
```

Se puede apreciar de forma clara que el contenido es poco intuitivo y difícil de seguir con los ojos en una lectura superflua. Se opta por aplicar el mismo procedimiento para ver los campos que componen a *routes*.

- Tras unos reajustes de variables reflejados en 3.7 se obtienen los siguientes campos *warnings, bounds, copyrights, overview polyline, legs, summary, waypoint order*

Código 3.7 Estudiando la estructura de routes.

```
// Routes commands
Cases[ Output_info[[2,2,1]],Rule[ x_,y_]->x]
```

De todos los campos obtenidos, *Legs* es el mas interesante puesto que contiene toda la información relacionada con el transporte, esto se puede verificar en [40] además de servir como consulta para mayor detalle sobre cualquiera de los campos obtenidos.

- Dentro de *Legs* siguiendo el proceso 3.8 se obtienen los nuevos campos sobre el transporte *departure time, duration, arrival time, end location, distance, end address, start address, start location, steps, traffic speed entry, via waypoint*. Aquí se tiene información precisa sobre las indicaciones que se siguen para llegar desde un punto a otro sobre la superficie terreste. Previamente, se obtuvo en *legs* las posibles rutas con las que *Google Maps API Directions* resuelve el problema de desplazamiento que se pide. Estas indicaciones son las asociadas a una de los posibles caminos para alcanzar el destino.

Código 3.8 Estudiando la estructura de legs.

```
// Legs commands
Output_info[[2,2,1,5,1]]
legs = Last[ legs /. Output_info[[2,2,1]]];
Cases[ legs,Rule[ x_,y_]->x]
```

- Dentro de las indicaciones, se estudia con las sentencias la nueva estructura interna de las mismas 3.9, conocidas como *steps*, obteniéndose *transit details, end location, distance, travel mode, start location, html instructions, duration, polyline*

Código 3.9 Estudiando la estructura de Steps.

```
// steps commands
trans= steps /. legs;
Map[Head,trans[[1]]]
Cases[trans[[1]],Rule[x_,y_]->x]
```

- Para encontrar la información de transporte público se debe acceder al campo *transit details* que contiene lo buscado. Como anteriormente, utilizando las sentencias 3.10 se obtiene *departure time, departure stop, arrival time, arrival stop, headsign, line, num stops*.

Código 3.10 Estudiando la estructura de Transit details.

```
//Transit_details commands
transi=transit_details /. trans;
Map[Head,transi[[1]]]
Cases[transi[[1]],Rule[x_,y_]->x]
```

Tras este proceso, los datos obtenidos son accesibles, se conoce la posición que ocupan, falta explicar como acceder a ellos. Utilizando las funciones *Position, ToExpression, ToString, StringForm, StringTake, StringDrop* se diseña una forma de acceso siguiendo el esquema de resultados sin alterar y aprueba de errores. Ejemplos de uso con estas funciones pueden ser consultados en [41].

3.3 *Mathematica* como herramienta para obtener los datos requeridos

Después de un análisis de funcionalidades utilizando *Mathematica*, se presentan como se extraen los datos de *Google Maps* para definir la red de transporte compuesta por nodos y enlaces o *links*.

3.3.1 Obtención de las redes de autobuses

Haciendo uso de las características de las peticiones previas, la extracción de las lineas de buses consta de 3 etapas:

1. Obtención de la polilínea a través de unas coordenadas dadas. Una polilinea se compone de un conjunto de coordenadas formando un trayecto de transporte.
2. Descomposición de la polilínea en sus unidades mas pequeñas extrayendo la información necesaria.
3. Organización de la información obtenida según el método de descomponer la polilinea.

En la primera fase descrita se obtienen los costes reales de transporte que *Google Maps* muestra cuando se accede a una consulta sobre transporte de un punto geográfico a otro, los nombres de las paradas, su localización en coordenadas de latitud longitud, la línea de bus asociada, su longitud en metros para un mapa a la hora que defina el usuario. Si se requiere la información en hora punta, se puede obtener pero en este caso se obtuvo la información en tiempo medio o genérico. Esta información se obtiene por medio de una petición única.

En la segunda fase se obtiene la información de las paradas que componen la polilinea mediante múltiples llamadas según describen los 2 códigos presentados a continuación. Los datos obtenidos son nombres de paradas y coordenadas concretas del nodo.

En la tercera fase se reorganiza la información citada obtenida para obtener un archivo *.wl* que contenga toda la información de una linea en una dirección concreta, sea ida o vuelta. Esta se compone de cada una de los nombres de las paradas ordenadas, las coordenadas y los tiempos entre parada y parada.

El código que describe las fases previamente explicadas es presentado ordenadamente en 3.11, 3.12,3.13, 3.14 y 3.15.

Código 3.11 Fase 1 - Obteniendo líneas.

```
Needs["GeneralUtilities‘"]

(* decoding the polyline *)
decodePolyline[encoded_String]:=Module[{deltas},deltas=With[{x=FromDigits
    [#,32]},If[OddQ[x],-BitShiftRight[x,1]-1,BitShiftRight[x,1]]]&/@Reverse[
    BitAnd[ToCharacterCode@StringCases[encoded,RegularExpression["[_-~]*[ -\\^]
    "]]-63,BitNot@32],2];
Rest@FoldList[Plus,{0,0},Partition[deltas/100000.,2]]]

LocNombreparada[lineagoogle_,final_:0]:=Module[{asroute,i=1,istep=2,nsteps,
    stPub,chosenstop="departure_stop"},
asroute=ToAssociations[lineagoogle];
nsteps=Length@asroute[["routes",1,"legs",1,"steps"]];
If[final>0,chosenstop="arrival_stop"];
stPub=Cases[ asroute[["routes",1,"legs",1,"steps"]],x_/;x[["travel_mode"]]=="
    TRANSIT"];
{
stPub[[1,"transit_details",chosenstop,"location","lat"]],
stPub[[1,"transit_details",chosenstop,"location","lng"]],
stPub[[1,"transit_details",chosenstop,"name"]]}
]
(* Google querys *)
peticionTransitGoogle[l_]:="https://maps.googleapis.com/maps/api/directions/
    json?origin="<> ToString[First@First@l]<>","<> ToString[Last@First@l]<>"&
    destination="<>ToString[First@Last@l]<>","<> ToString[Last@Last@l]<>"&mode=
    transit"

peticionTransitGoogle2[l_]:="https://maps.googleapis.com/maps/api/directions/
    json?origin="<> ToString[First@First@l]<>","<> ToString[Last@First@l]<>"&
    destination="<>ToString[First@Last@l]<>","<> ToString[Last@Last@l]<>"&mode=
    transit"<>"&alternatives=true"<>"&types=bus_station"

peticionTransitGoogle3[l_]:="https://maps.googleapis.com/maps/api/directions/
    json?origin="<> ToString[First@First@l]<>","<> ToString[Last@First@l]<>"&
    destination="<>ToString[First@Last@l]<>","<> ToString[Last@Last@l]<>"&mode=
    transit"<>"&alternatives=true"

peticionTransitGoogleStatio[l_]:="https://maps.googleapis.com/maps/api/place/
    nearbysearch/json?location="<> ToString[First@First@l]<>","<> ToString[
    Last@First@l]<>"&sensor=true"<>"&rankby=1"<>"&types=transit_station"<>"&
    transit_routing_preference=less_walking"

peticionTransitGooglelcortas[l_]:="https://maps.googleapis.com/maps/api/
    directions/json?origin="<> ToString[First@First@l]<>","<> ToString[
    Last@First@l]<>"&destination="<>ToString[First@Last@l]<>","<> ToString[
```

```
    Last@Last@l]<>"&key=AIzaSyDrfqhAAFFPib6OMOill-qv9Y7MRn_xuIA"<>"&mode=
    transit"<>"&alternatives=true"<>"&transit_mode=bus"<>"&
    transit_routing_preference=less_walking"

peticionTransitGoogle5[l_]:="https://maps.googleapis.com/maps/api/directions/
    json?origin="<> ToString[First@First@l]<>","<> ToString[Last@First@l]<>"&
    destination="<>ToString[First@Last@l]<>","<> ToString[Last@Last@l]<>"&key=
    AIzaSyDrfqhAAFFPib6OMOill-qv9Y7MRn_xuIA"<>"&mode=transit"<>"&transit_mode=
    bus"

peticionTransitGoogle6[l_]:="https://maps.googleapis.com/maps/api/directions/
    json?origin="<> ToString[First@First@l]<>","<> ToString[Last@First@l]<>"&
    destination="<>ToString[First@Last@l]<>","<> ToString[Last@Last@l]<>"&key=
    AIzaSyDrfqhAAFFPib6OMOill-qv9Y7MRn_xuIA"<>"&mode=transit"<>"&transit_mode=
    bus"<>"&transit_routing_preference=less_walking"

peticionTransitGoogle7[l_]:="https://maps.googleapis.com/maps/api/directions/
    json?origin="<> ToString[First@First@l]<>","<> ToString[Last@First@l]<>"&
    destination="<>ToString[First@Last@l]<>","<> ToString[Last@Last@l]<>"&key=
    AIzaSyDrfqhAAFFPib6OMOill-qv9Y7MRn_xuIA"<>"&mode=transit"<>"&transit_mode=
    bus"<>"&transit_routing_preference=less_walking"<>"&traffic_model=
    pessimistic"

peticionTransitGoogle8[l_]:="https://maps.googleapis.com/maps/api/directions/
    json?origin="<> ToString[First@First@l]<>","<> ToString[Last@First@l]<>"&
    destination="<>ToString[First@Last@l]<>","<> ToString[Last@Last@l]<>"&key=
    AIzaSyDrfqhAAFFPib6OMOill-qv9Y7MRn_xuIA"<>"&mode=transit"<>"&departure_time
    =now"<>"&language=en"<>"&traffic_model=pessimistic"

peticionTransitGooglelcortashour[l_,times_]:="https://maps.googleapis.com/maps/
    api/directions/json?origin="<> ToString[First@First@l]<>","<> ToString[
    Last@First@l]<>"&destination="<>ToString[First@Last@l]<>","<> ToString[
    Last@Last@l]<>"&key=AIzaSyDrfqhAAFFPib6OMOill-qv9Y7MRn_xuIA"<>"&mode=
    transit"<>"&alternatives=true"<>"&transit_routing_preference=less_walking"
    <>"&departure_time="<>ToString[times]

// FASE 1

(* lat and long and other necessary variables*)
l = {{37.360222,-5.9731827},{37.3925383,-5.9952423}}

(* line 1 {{37.411145, -5.983429},{37.3572018,-5.9804376}}
line 2 {{37.3558741,-5.9867478 },{37.4038631,-5.9943069}}
line 3 {{37.426399, -5.964823},{37.314271, -5.970962}}
line 5 {{37.3867428,-5.9512907},{37.3909354,-6.0104641}}
line 6 {{37.4130247,-5.986079},{37.3571232,-5.9823351}}
line 10 {{37.424907,-5.98269},{37.3933408,-5.9877}}
line 11 {{37.393292, -5.987513},{37.41259,-5.9744382}}
line 12 {{37.3931347,-5.9875592},{37.422437,-5.9664033}}
line 13 {{37.393142, -5.995704},{37.432220, -5.970801}}
line 14  {{37.4096148,-5.9844714},{37.392919, -5.995717}}
line 15 {{37.4116204,-5.9679141},{37.3929581,-5.9867875}}
line 16 {{37.4304667,-5.9268443},{37.3929758,-5.9868252}}
line 20  {{37.4002798,-5.957256},{37.3932738,-5.9877222}}
line 21 {{37.3999946,-5.9569233},{37.391877, -6.003508}}
line 22 {{37.4063019,-5.9133368},{37.3802474,-5.9855484}}
line 24  {{37.393392, -5.987599},{37.3750069,-5.9425053}}
```

```
line 25   {{37.3749894,-5.9479795},{37.3804841,-5.9862721}}
line 26 {{37.380455, -5.985876},{37.3659313,-5.9496413}}
line 27 {{37.388828,-5.9227625},{37.392712,-5.9973269}}
line 28 {{37.3802098,-5.9860367},{37.4100761,-5.9262477}}
    {{37.3803105,-5.9853172},{37.410238,+-5.926136}} check app tom
line 29 {{37.382513, -5.905396},{37.380238, -5.985024}}
line 30 {{37.380124, -5.985063},{37.3620278,-5.9606133}}
line 31 {{37.380179, -5.985301},{37.3609117,-5.9657953}}
line 32 {{37.360222,-5.9731827},{37.3925383,-5.9952423}}
line 34 {{37.3800906,-5.9887932},{37.3482879,-5.9797056}}
line 37 {{37.380731, -5.993979},{37.3142865,-5.971097}}
line 38A {{37.380399, -5.985661},{37.3463456,-5.9483517}}
line 39 {{37.387857, -5.958612},{37.3756049,-5.9302546}}
line 40 {{37.3909378,-5.9970624},{37.3770204,-6.0128807}}
line 41 {{37.368319,-6.0116799},{37.3906225,-5.9973553}}
line 43 {{37.3910993,-5.9973247},{37.3761484,-6.0140253}}
line 52 {{37.3781708,-5.9796367},{37.3751814,-5.9302616}}
line B3 {{37.400838, -5.947224},{37.3816375,-5.9661688}}
line EA {{37.3916077,-6.0040885},{37.4234812,-5.9004222}} *)

buslchos="32"
mode="ida"
Namefil = "raw_linea_"<>ToString[buslchos]<>"_"<>ToString[mode]<>".wl"
Namefin = "linea_"<>ToString[buslchos]<>"_"<>ToString[mode]<>".wl"

(*  Google query *)
resp = URLExecute[peticionTransitGooglelcortas[1]];

(* extracting info from the first field*)
respuestaslinea={};
AppendTo[respuestaslinea,resp[[1]]];

(* Detecting the correct line *)
posilines = Position[resp,"short_name"];

(* obtention of the index of the matrix by means of StringDrop[StringTake[
    Tostring[posilines[[]]],38],1]*)
tesval = "00" ;
i=1;
While[tesval!=buslchos,
str = StringDrop[StringTake[ToString[posilines[[i]]],StringLength[ToString[
    posilines[[i]]]]-2],1]<>"2";
If[ToExpression[ToString[StringForm["resp[[``]]",str]]]==buslchos,tesval=
    buslchos,Print["Not the line"]];
i++;
];

(* Selecting the necessary information *)
keyindex = i - 1;
resp2 = resp[[2,2,keyindex]];

(* dedocding the polyline ----  POSSIBLE UNDERSTANDER*)
transnes = Position[resp2,"polyline"];
(* character counter determiner *)
varad = 0;strvar ={};
For[u=1,u<=Length@transnes,u++,
```

```
transnesind = StringDrop[StringTake[ToString[transnes[[u]]],StringLength[
    ToString[transnes[[u]]]]-2],1]<>"2";
newvar = ToExpression[ToString[StringForm["resp2[[‘‘]]",transnesind]]];
polpos = Position[newvar,"points"];
puntosind = StringDrop[StringTake[ToString[polpos[[1]]],StringLength[ToString[
    polpos[[1]]]]-2],1]<>"2";
puntos = ToExpression[ToString[StringForm["newvar[[‘‘]]",puntosind]]];
If[varad<StringLength[ToString[puntos]],varad = StringLength[ToString[puntos]];
     Clear[strvar]; strvar = puntos];
];
polilinea = decodePolyline[strvar]
```

Código 3.12 Fase 2 a - Descomposición de polilinea.

```
(* Polyline per step USING TRANSIT 4 AND KEY*)

(* setting the directory desired *)
If[!DirectoryQ[FileNameJoin[{NotebookDirectory[],"prueba"}]],
disit = FileNameJoin[{NotebookDirectory[],"prueba"}];
dir=CreateDirectory[disit];SetDirectory[dir];
CreateFile[ToString[StringForm["‘‘",Namefil]]]
,Print["ya existe tronco"]];

(* saving errors *)
coorerr={};

(* Stops and lat long *)
For[i=0,i<=Length@polilinea-2,i++,
coor={polilinea[[1]],polilinea[[i+2]]};
infourl = URLExecute[peticionTransitGooglelcortas[coor]];
(* infourl = URLExecute[peticionTransitGooglelcortashour[coor,timm]]; *)
(* Checking bus line correctness*)
If[i==0,coor={polilinea[[2]],polilinea[[Length@polilinea]]};
infourl = URLExecute[peticionTransitGooglelcortas[coor]];]; (* Avoiding first
    query error *)
tesval = "00" ; j=1; auxlin = Position[infourl,"short_name"];
While[tesval!=buslchos,
str = StringDrop[StringTake[ToString[auxlin[[j]]],StringLength[ToString[auxlin
    [[j]]]]-2],1]<>"2";
Which[ToExpression[ToString[StringForm["infourl[[‘‘]]",str]]]==buslchos,tesval=
    buslchos,
 ToExpression[ToString[StringForm["infourl[[‘‘]]",str]]]!=buslchos,Print["Not
     the line"],
 j>Length@polilinea,{Print["line not found; Coordinates saved"]; AppendTo[
     coorerr,1];Break[]}];
j++;
];
keypar=j-1;
Savvar = infourl[[2,2,keypar]];
AppendTo[respuestaslinea,Savvar];
(*Avoiding Google stop of queryings Pause[0.05];*)
Pause[0.05];
]
```

Código 3.13 Fase 2 b - Descomposición de polilinea.

```
(* Polyline per step WITHOUT IT*)
respuestaslinea={};
(* setting the directory desired *)
If[!DirectoryQ[FileNameJoin[{NotebookDirectory[],"prueba"}]],
disit = FileNameJoin[{NotebookDirectory[],"prueba"}];
dir=CreateDirectory[disit];SetDirectory[dir];
CreateFile[ToString[StringForm["``",Namefil]]]
,Print["ya existe tronco"]];

(* saving errors *)
coorerr={};

(* Stops and lat long *)
For[i=1,i<=Length@polilinea/2,i++,
coor={polilinea[[i]],polilinea[[Length@polilinea]]};
infourl = URLExecute[peticionTransitGooglelcortas[coor]];
(* Checking bus line correctness*)
tesval = "00" ; j=1; auxlin = Position[infourl,"short_name"];
While[tesval!=buslchos,
str = StringDrop[StringTake[ToString[auxlin[[j]]],StringLength[ToString[auxlin
    [[j]]]]-2],1]<>"2";
Which[ToExpression[ToString[StringForm["infourl[[``]]",str]]]==buslchos,tesval=
    buslchos,
 ToExpression[ToString[StringForm["infourl[[``]]",str]]]!=buslchos,Print["Not
     the line"],
 j>Length@polilinea,{Print["line not found; Coordinates saved"]; AppendTo[
     coorerr,1]}];
j++;
];
keypar=j-1;
Savvar = infourl[[2,2,keypar]];
AppendTo[respuestaslinea,Savvar];
(*Avoiding Google stop of queryings*)
Pause[0.05];
];
fact1=i-1;
For[i=Floor[Length@polilinea/2]+1,i<=Length@polilinea,i++,
coor={polilinea[[1]],polilinea[[i]]};
infourl = URLExecute[peticionTransitGoogle3[coor]];
(* Checking bus line correctness*)
tesval = "00" ; j=1; auxlin = Position[infourl,"short_name"];
While[tesval!=buslchos,
str = StringDrop[StringTake[ToString[auxlin[[j]]],StringLength[ToString[auxlin
    [[j]]]]-2],1]<>"2";
Which[ToExpression[ToString[StringForm["infourl[[``]]",str]]]==buslchos,tesval=
    buslchos,
 ToExpression[ToString[StringForm["infourl[[``]]",str]]]!=buslchos,Print["Not
     the line"],
 j>Length@polilinea,{Print["line not found; Coordinates saved"]; AppendTo[
     coorerr,1]}];
j++;
];
keypar=j-1;
Savvar = infourl[[2,2,keypar]];
AppendTo[respuestaslinea,Savvar];
```

```
(*Avoiding Google stop of queryings*)
Pause[0.05];
];
fact2=i-Floor[Length@polilinea/2]+1-1;
```

Código 3.14 Fase 3 a - Reorganización de paradas y guardado de infromación.

```
strf[info_,order_]:=StringDrop[StringTake[ToString[info[[order]]],StringLength[
    ToString[info[[order]]]]-2],1]<>"2"<>",2"<>",2"
// Fase 3 A
 (* Preparing line array*)
linearray={};

(* Extrating information of first*)
Salidas = Position[respuestaslinea,"departure_stop"];
str = strf[Salidas,2];
AppendTo[linearray,{polilinea[[1]][[1]],polilinea[[1]][[2]],
ToExpression[ToString[StringForm["respuestaslinea[[``]]",str]]]}];

Salidas = Position[respuestaslinea,"departure_stop"];
str = strf[Salidas,1];
AppendTo[linearray,{polilinea[[2]][[1]],polilinea[[2]][[2]],
ToExpression[ToString[StringForm["respuestaslinea[[``]]",str]]]}];

(* arrivals *)
Llegadas = Position[respuestaslinea,"arrival_stop"];
For[t=2,t<=Length@Llegadas,t++,
str = strf[Llegadas,t];
AppendTo[linearray,{polilinea[[t+1]][[1]],polilinea[[t+1]][[2]],
ToExpression[ToString[StringForm["respuestaslinea[[``]]",str]]]}];
]
```

Código 3.15 Fase 3 b - Reorganización de paradas y guardado de información.

```
strf[info_,order_]:=StringDrop[StringTake[ToString[info[[order]]],StringLength[
    ToString[info[[order]]]]-2],1]<>"2"<>",2"<>",2"
// Fase 3 B
(* Preparing line array*)
linearray={};

Salidas = Position[respuestaslinea,"departure_stop"];
For[t=1,t<=fact1,t++,
str = strf[Salidas,t];
AppendTo[linearray,{polilinea[[t]][[1]],polilinea[[t]][[2]],
ToExpression[ToString[StringForm["respuestaslinea[[``]]",str]]]}];
]

(* arrivals *)
Llegadas = Position[respuestaslinea,"arrival_stop"];
For[t=fact2,t<=Length@polilinea,t++,
str = strf[Llegadas,t];
AppendTo[linearray,{polilinea[[t]][[1]],polilinea[[t]][[2]],
ToExpression[ToString[StringForm["respuestaslinea[[``]]",str]]]}];
]
```

Un ejemplo de la información obtenida en bruto se muestra en 3.6. Tras el filtro aplicado en 3.15 y 3.14, se obtienen unos resultados como en 3.16

Código 3.16 Información filtrada.

```
{{37.4111, -5.98342,
  "Avenida San Lázaro (Doctor Morote)"}, {37.4123, -5.98237,
  "Trabaj. Inmigrantes(Diego de Almagro)"}, {37.411, -5.98091,
  "Trabaj. Inmigrantes (Diego Puerta)"}, {37.4085, -5.9798,
  "Trabaj. Inmigrantes (Los Romeros)"}, {37.407, -5.97955,
  "Madreselva"}, {37.4048, -5.98053,
  "Doctor Jiménez Díaz"}, {37.4029, -5.98139,
  "Avenida La Cruz Roja (Antonio Machín)"}, {37.4003, -5.98268,
  "Avenida La Cruz Roja (Clínica)"}, {37.3977, -5.9815,
  "Ronda de Capuchinos (La Trinidad)"}, {37.3942, -5.98386,
  "María Auxiliadora (Puerta Osario)"}, {37.3913, -5.98424,
  "Recaredo (San Roque)"}, {37.3884, -5.98499,
  "Menéndez Pelayo (Puerta Carmona)"}, {37.3862, -5.98602,
  "Menéndez Pelayo (Puerta de La Carne)"}, {37.3803, -5.98595,
  "Avenida Carlos V (Prado San Sebastián)"}, {37.3774, -5.98424,
  "Avenida La Borbolla (Capitanía)"}, {37.3743, -5.98542,
  "Avenida La Borbolla (Montevideo)"}, {37.3724, -5.98562,
  "Avenida La Borbolla (Plaza América)"}, {37.3676, -5.98532,
  "Manuel Siurot (Edificio La Estrella)"}, {37.3642, -5.98342,
  "Manuel Siurot (Cardenal Ilundain)"}, {37.3613, -5.98188,
  "Manuel Siurot (Hospital V.Rocio)"}, {37.3595, -5.98084,
  "Manuel Siurot (Francisco Murillo)"}, {37.3574, -5.97977,
  "Manuel Siurot (Carretera Su Eminencia)"}, {37.3567, -5.97564,
  "Ctra. Su Eminencia (Castillo de Marcheni"}, {37.357, -5.98032,
  "Glorieta Plus Ultra"}}
```

3.3.2 Obtención del modo a pie

Para el modo a pie se necesita primero describir como están organizadas las diferentes zonas de transporte para mostrar los flujos de viajeros. Un mapa de los centroides de distribución de viajeros queda descrito en 3.5.2.

A continuación, la información que se extrae es el tiempo a pie desde los centroides presentados a cada una de las paradas del transporte público previamente obtenidas. Se hace considerando si las paradas de todas las lineas se encuentran a menos de un radio de 500 metros del centroide estudiado. El código que realiza esta operación queda descrito en 3.17.

Código 3.17 Obtención de los *links* del modo a pie.

```
(* Now using the centroids and the available stops check the belonging *)
AbsoluteTiming[
walkinglinkaux = {};
SetSharedVariable[walkinglink,walkinglinkaux,binarybus,binarybusaux];
Parallelize[
i=164;
Do[
binarybusaux={};
Do[
reg = GeoDisk[GeoPosition[{Centroids[[i,1,1]],Centroids[[i,1,2]]}],Rext];
If[GeoWithinQ[reg,GeoPosition[{Lineas[[j,k,1]],Lineas[[j,k,2]]}]],
(* Query google walking mode*)
```

```
(* Filling final arrays: ID \[Rule] bus stop + time and viceversa, binary
    matrix with go and return
to check where we should divide the bus lines
division is donde using constant and length of paths
After finishing the division mode it is time to create the final map *)
(* get the information from query*)
l = {{Centroids[[i,1,1]],Centroids[[i,1,2]]},{Lineas[[j,k,1]],Lineas[[j,k
    ,2]]}};
resp = URLExecute[peticionTransitGoogle7[l]];
If[resp[[3,2]]=="OK",
(* info needed*)
Duraa = Position[resp,"duration"];
Dista = Position[resp,"distance"];
name = Centroids[[i,2]]<>"_"<>ToString[Lineas[[j,k,3]]];
distacc = ToExpression[ToString[StringForm["resp[[‘‘]]",strf1b[Dista,1]]]];
duracc = ToExpression[ToString[StringForm["resp[[‘‘]]",strf1b[Duraa,1]]]];
AppendTo[walkinglinkaux,{{name},duracc,l,distacc,j}];
(*inverse *)
name = ToString[Lineas[[j,k,3]]]<>"_"<>Centroids[[i,2]];
l = {{Lineas[[j,k,1]],Lineas[[j,k,2]]},{Centroids[[i,1,1]],Centroids[[i
    ,1,2]]}};
AppendTo[walkinglinkaux,{{name},duracc,l,distacc,j}];
(* dividing *)
AppendTo[binarybusaux,1];];
,
(*No case fill the binary bus*)
AppendTo[binarybusaux,0];];
,{k,Range[Length@Lineas[[j]]]};
AppendTo[binarybus,{binarybusaux}];
Print[j]
,{j,Range[Length@Lineas]};
AppendTo[walkinglink,walkinglinkaux];];]
```

Se comprobó que el software *Mathematica* se puede utilizar para deducir el mismo tipo de resultados en cuanto al modo walking con una diferencia fundamental, los datos de *Google Maps* pueden corresponder a diferentes situaciones de tráfico, verse afectados por la congestión del momento, recalcularse y reactualizarse. Esta es la principal razón por la que se utilizaron esos datos.

Un ejemplo de camino a pie e puede apreciar en 3.1.

3.3.3 Obtención de los transbordos

Faltaría considerar la posibilidad de que un viajero cambie de línea sin pasar por un centroide. Esto se hace comprobando para cada parada si hay otra parada cerca a un radio de 200 metros, excluyendo la propia línea en la que circula en su dirección como en la contraria. Se penaliza la selección de los transbordos con un factor de un 1.2 con el fin de obligar al viajero supuesto a seguir un comportamiento más natural durante el viaje. El código encargado de esta labor se muestra en 3.18.

Código 3.18 Obtención de los *links* de transbordos.

```
(* TRANSBORD - First part checking part of the lines inside of the 200 m circle
    *)
lista = Range[Length@Lineas];
AbsoluteTiming[
SetSharedVariable[Lingrup,i,j,k,Indiceses,transb];
Parallelize[
transb={};
Indiceses={};
```

Figura 3.1 Camino a pie (azul) frente a camino en vehículo privado. Se escenifica la diferencia entre andar y conducir para la resolución del problema de asignación. Realizado con centroide 164 (situado en la parte mas sur de los considerados en este proyecto) y la parada *Ctra. Isla Menor (Inst. La Grasa)*.

```
Do[
Do[
If[OddQ[i],derry = Delete[lista,{{i},{i+1}}],derry = Delete[lista,{{i-1},{i
    }}]];
Do[
Lingrup = Reap[
```

```
(* do de lineas*)
Do[Sow[GeoPosition[{Lineas[[k,1,1]],Lineas[[k,1,2]]}]],{l,Range[Length@Lineas[[
    k]]]}]][[2]];
indesss = Position[GeoWithinQ[GeoDisk[GeoPosition[{Lineas[[i,j,1]],Lineas[[i,j
    ,2]]}],Rext],Lingrup],True];
(* zona de almacenaje *)
If[Length@indesss!=0,
(* Ldefinit = Delete[Flatten[indesss],1];*)
Ldefinit = Take[Flatten[indesss],{2,Length@Flatten[indesss],2}];
(*do de indices*)
Indiceses= Reap[Do[Sow[{i,j,k,be}],{be,Ldefinit}]][[2]];
AppendTo[transb,Indiceses];];
,{k,derry}];
(*AppendTo[transb,{alter}];*)
,{j,Range[Length@Lineas[[i]]]}]
,{i,Range[5,22]}]]]
```

Como ejemplo de transbordo se propone el escenificado en 3.2.

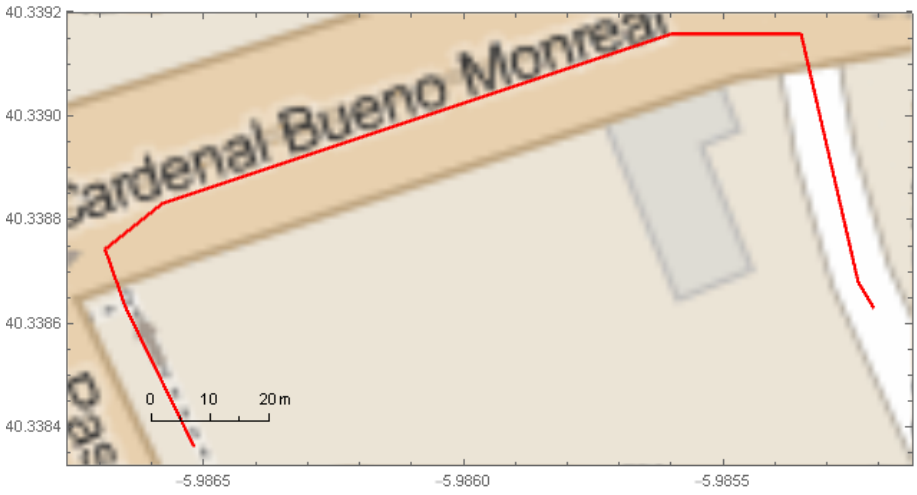

Figura 3.2 Ejemplo de transbordo a pie entre 2 paradas a menos de 200 metros. Las paradas de derecha a izquierda de la imagen son *La Palmera (Bueno Monreal)* y *Manuel Siurot (Edificio La Estrella)* respectivamente.

3.3.4 Uniendo la información para ser tratada en *Matlab*

Previo a introducir la información en *Matlab*, la información de todo el mapa sigue un formato claro y determinado descrito en 3.1 3.2 para la identificación de link o nodo correspondientemente. Para el caso de las paradas de bus 2 modos fueron introducidos para representar las opciones de comportamiento que podría tener un viajero cualquiera. Por un lado, el *link* como viaje completo (desde su parada inicial a la final) caracterizado por su frecuencia y el coste completo. Por otro, estableciendo enlaces entre paradas consecutivas caracterizado cada enlace por su frecuencia y el tiempo dependiente de la distancia (bajo el mapa de *Google Maps*) del enlace entre paradas calculado proporcionalmente con respecto a la distancia total. Se adjuntan los códigos utilizados en el filtrado en 3.19 y 3.20

Código 3.19 Filtrado de *links*.

```
// Preámbulos
files=FileNames["*.wl","C:\\Users\\Javier\\Desktop\\TFM sevilla luis\\database
    controll\\calculos walking ID"];
Dimensions[files];
Walkingbars = Import[#]&/@files;
```

```
(* Function for number of transit *)
sizet = 13;
sizedec = 11;

nametrstp[par1_]:="la"<>StringDelete[ToString[NumberForm[
First@First@par1,{sizet,sizedec}]],{".","-"}]<>"lo"<>StringDelete[ToString[
    NumberForm[
Last@First@par1,{sizet,sizedec}]],{".","-"}]<>"_"<>"la"<>StringDelete[ToString
[NumberForm[First@Last@par1,{sizet,sizedec}]],{".","-"}]<>"lo"<>StringDelete[
ToString[NumberForm[Last@Last@par1,{sizet,sizedec}]],{".","-"}]

(* nametrstp[par1_]:=StringDelete[ToString[NumberForm[First@First@par1,{sizet,
    sizedec}]]
,{".","-"}]<>StringDelete[ToString[NumberForm[Last@First@par1,{sizet,sizedec}]]
,{".","-"}]<>"_"<>StringDelete[ToString[NumberForm[First@Last@par1,
{sizet,sizedec}]],{".","-"}]<>StringDelete[ToString[NumberForm[Last@Last@par1,
{sizet,sizedec}]],{".","-"}] *)

nametrstpUno[par1_]:="la"<>StringDelete[ToString[NumberForm[
First@par1,{sizet,sizedec}]],{".","-"}]<>"lo"<>StringDelete[ToString[NumberForm
    [
Last@par1,{sizet,sizedec}]],{".","-"}]

(* nametrstpUno[par1_]:=StringDelete[ToString[NumberForm[First@par1,{sizet,
    sizedec}]],
{".","-"}]<>StringDelete[ToString[NumberForm[Last@par1,{sizet,sizedec
    }]],{".","-"}] *)


// Bus
Buslink ={};
Buslinkcom = {};
jj = 0;
Do[
If[OddQ[i],aa=5;jj++,aa = 6;]; (* esto controla vuleta e ida*)
Do[
If[j==Length@Lineas[[i]],
name = Lineas[[i,1,3]]<>"_"<>Lineas[[i,j,3]];
lanam = {{Lineas[[i,1,1]],Lineas[[i,1,2]]},{Lineas[[i,j,1]],Lineas[[i,j,2]]}};
AppendTo[Buslink,{nametrstp[lanam],Frecuencia[[jj,aa]],Frecuencia[[jj,4]],
    ToString[name]}];
AppendTo[Buslinkcom,{ToString[name],{},Frecuencia[[jj,aa]],Frecuencia[[jj
    ,4]]}];,
(* array transit stop to transit stop*)
Distanss = QuantityMagnitude[UnitConvert[GeoDistance[{Lineas[[i,j,1]],Lineas[[i
    ,j,2]]},{Lineas[[i,j+1,1]],Lineas[[i,j+1,2]]}],"Kilometers"]];
name = Lineas[[i,j,3]]<>"_"<>Lineas[[i,j+1,3]];
lanam = {{Lineas[[i,j,1]],Lineas[[i,j,2]]},{Lineas[[i,j+1,1]],Lineas[[i,j
    +1,2]]}};
AppendTo[Buslink,{nametrstp[lanam],N[Frecuencia[[jj,aa]]*Distanss/Frecuencia[[
    jj,aa+2]]],Frecuencia[[jj,4]],ToString[name]}];
AppendTo[Buslinkcom,{ToString[name],{},N[Frecuencia[[jj,aa]]*Distanss/
    Frecuencia[[jj,aa+2]]],Frecuencia[[jj,4]]}];];
,{j,Range[Length@Lineas[[i]]]}]
,{i,Range[Length@Lineas]}]
```

```
// A pie
Walkending = {};
Walkenmatlab = {};
Do[
Do[
Do[
(* primer formato que no funcionó
AppendTo[Walkending,{Walkingbars[[i,j,k,1,1]],Walkingbars[[i,j,k,3]],
N[Walkingbars[[i,j,k,2,2,2]]/60],"inf"}];
AppendTo[Walkenmatlab,{Walkingbars[[i,j,k,1,1]],Walkingbars[[i,j,k,3]],
N[Walkingbars[[i,j,k,2,2,2]]/60],"inf"}]; *)
AppendTo[Walkending,{nametrstp[Walkingbars[[i,j,k,3]]],
N[Walkingbars[[i,j,k,2,2,2]]/60],"inf",Walkingbars[[i,j,k,1,1]]}];
,{k,Range[Length@Walkingbars[[i,j]]]}]
,{j,Range[Length@Walkingbars[[i]]]}]
,{i,Range[Length@Walkingbars]}]

// Links finales
AllLinks = Join[Buslink,Walkending];
tmp = FileNameJoin[{NotebookDirectory[],"lineas_new","All_Links.xlsx"}];
Export[tmp,AllLinks]
```

Código 3.20 Filtrado de nodos.

```
// Centroides

Centroides={};
Centroidesall={};
Do[
AppendTo[Centroides,{nametrstpUno[Centroids[[i,1]]],0,0,Centroids[[i,2]]}];
,{i,Range[Length@Centroids]}]

//Paradas
Nodelin={};
Nodelinall={};
Do[
Do[
AppendTo[Nodelin,{nametrstpUno[{Lineas[[i,j,1]],Lineas[[i,j,2]]}],0,0,ToString[
    Lineas[[i,j,3]]]}];
,{j,Range[Length@Lineas[[i]]]}]
,{i,Range[Length@Lineas]}]

// Unir y guardar
AllNodes = Join[Nodelin,Centroides];
tmp = FileNameJoin[{NotebookDirectory[],"lineas_new","Total_nodes.xlsx"}];
Export[tmp,AllNodes]
```

Código 3.21 Filtrado *links* de transbordos.

```
(* Walking time extraction *)
tranbis = Flatten[transb,1];
AbsoluteTiming[
SetSharedVariable[Pepeoso];
Parallelize[
Pepeoso={};
```

```
Pepeoso = Reap[Do[
i=tranbis[[d,1]];
j= tranbis[[d,2]];
k = tranbis[[d,3]];
be = tranbis[[d,4]];
(* Info to extract *)
Sow[{{nametrstp[{{Lineas[[i,j,1]],Lineas[[i,j,2]]},{Lineas[[k,be,1]],Lineas[[k,
    be,2]]}}]},
QuantityMagnitude[UnitConvert[TravelTime[{GeoPosition[{Lineas[[i,j,1]],Lineas[[
    i,j,2]]}],
GeoPosition[{Lineas[[k,be,1]],Lineas[[k,be,2]]}]},TravelMethod -> "Walking"],"
    Minutes"]],"inf"}];
,{d,Range[Length@Flatten[transb,1]]}]][[2]];]]

(* Deleting wrong responses*)
tranbis = Flatten[transb,1];
AbsoluteTiming[
testeeee = Reap[Do[
If[NumberQ[Pepeoso[[dueto,2]]],Sow[Pepeoso[[dueto]]]];
,{dueto,Range[Length@Pepeoso]}]][[2]]];

( * Exporting final info to xlsx * )
endlist=Flatten[Join[transbordis[[1]],PePeosi],1]
tmp = FileNameJoin[{NotebookDirectory[],"lineas_new","Total_links_end.xlsx"}];
Export[tmp,endlist]
```

Se obtuvieron 2 listas cuyos conjuntos estaban compuestos por el completo de todos los nodos obtenidos y todos los enlaces respectivamente. Los códigos utilizados para generar y gestionar esta información quedan descritos en 3.19, 3.20 y 3.21. De entre todos los problemas, el más acentuado fue que las rutinas que comprobaban parada a parada la inclusión de la misma dentro del radio de 200 para considerarse transbordo requiere de la función *GeoWithinQ* de *Mathematica*. Esta es lenta en ejecución, no se trata de ninguna entidad *builtin* ni similares. La solución tomada es un cambio de enfoque en 2 aspectos principales de la metodología diseñada:

1. Se crea la entidad línea como un conjunto de paradas discretas pero asociadas como una grupo para comprobar la pertenencia del mismo al radio exigido en lugar de parada a parada dentro del marco 3.21
2. Se utilizan sus índices generados dentro de los bucles anidados para hacer una nueva operación de llamada obteniendo los tiempos de transporte entre los puntos que cumplan estas condiciones.

Con estas soluciones, se resuelven los problemas de velocidad experimentados. Quizás no a unos valores óptimos pero si a un rango aceptable de espera (en total menos 3 horas para todos los datos) paralelizando las operaciones. Además, se debe comentar que podrían ser mucho mas rápidos pero eso influiría muy *negativamente* en el *bolsillo* del autor del proyecto puesto que *Google Maps* limita las llamadas por día a 2500 en modo a pie. Una tarifa distinta de la gratuita implicaría un desembolso superior a los 90 euros. Por otro lado, se ha garantizado y comprobado que los tiempos de transbordo obtenidos con *Mathematica* mediante *TravelTime* y *Google Maps* son los mismos disipando cualquier tipo de duda sobre la veracidad, obtención o calidad de los resultados presentados.

Matlab por otro lado se encarga de eliminar redundancias en las listas de nodos y enlaces mediante un filtrado como el que se expresa en 3.22. Se trata de eliminar las incoherencias de los datos tras su exportación de una plataforma a otra. Los problemas que se encontraron fueron:

- Error al reconocer la entidad *Inf* propuesta por el *IEEE* para la identificación de esta entidad matemática en el paso de *Mathematica* a *Matlab*. La solución prevista revisa cada uno de los enlaces y sus valores para no dejar ningún incoherencia que pudiese manifestarse como tipo *NaN*. El problema surge de un malentendido entre las plataformas para intercambiar los datos por el tipo de datos utilizados.
- Error por repetición de *links* o *nodos* cuando los cálculos se ejecutan. La solución es comprobar mediante *unique* de *Matlab* la unicidad de los datos evitando elegantemente este problema.

Código 3.22 Filtrado de redundancias en *links* utilizando *Matlab*.

```
%% Importing Mathematica information
tic
% NODES - Every node used is loaded here
[~,txt,~] = xlsread('C:\Users\Javier\Desktop\TFM sevilla luis\database
    controll\End database\Nodes\Total_nodes.xls');
% however this data need to be reorganized. There is no more info needed
% but in this case we only are going to use the TXT info
[Unico,ca,~] = unique(txt(:,1)); % elimination of ijntersecting nodes
% using the same values positionet at the same row
var = txt(:,2);
var2 = txt(:,3);
var3 = txt(:,4);

var = var(ca);
var2 = var2(ca);
var3 = var3(ca);

nodes = cell2struct([var var2 var3],Unico,1);
nodes = nodes';

% filling with adequate info
Rnod = Centroids{parad,1}; % destination node
fild = fieldnames(nodes);
for i = 1:length(fild)
    if strcmp(fild{i},Rnod)
        nodes(1).(sprintf('%s',fild{i})) = 0;% using sprintf we can access
            to the data using the name
        nodes(2).(sprintf('%s',fild{i})) = 0;
        disp('tienes nodo')
    else
        nodes(1).(sprintf('%s',fild{i})) = inf;
        nodes(2).(sprintf('%s',fild{i})) = 0;
    end
end

%struct2table(nodes)

clear var var2 var3 num txt raw Unico

% LINKS
[num,txt,raw] = xlsread('C:\Users\Javier\Desktop\TFM sevilla luis\database
    controll\End database\Links\Total_links_end.xlsx');

[Unico,ca,ci] = unique(txt(:,1)); % elimination of ijntersecting nodes
% using the same values positionet at the same row
var = num2cell(num(:,1));
var2 = num(:,2);

% treament for POSSIBLE Nan
indisess = isnan(var2);

var2(indisess) = Inf;
var2 = num2cell(var2);
```

```
    var3 = txt(:,4);

    var = var(ca);
    var2 = var2(ca);
    var3 = var3(ca);

    links = cell2struct([var var2 var3],Unico,1);

    links = links';
```

Acto seguido, se presentan los datos obtenidos para esta red de transporte utilizando diversas zonas del mapa que contiene a la ciudad de Sevilla de esta forma se aumenta considerablemente la calidad visual y se resumen la información dando una perspectiva única y general sobre el trabajo realizado.

3.4 Definiendo una red genérica de transporte público

Las red de transporte utilizada queda definida de la siguiente forma:

- Nodos. Todas las paradas y centroides considerados.
- Enlaces. El conjunto de transbordos, mapa a pie de centroides a paradas de bus y los de las lineas de autobús.

La forma en la que se almacenan se define a posteriori para presentar después los datos utilizados y obtenidos durante las peticiones de información.

Las herramientas de análisis y los parámetros de diseño que permiten una correcta asignación del medio público en nuestra red se pueden encontrar definidas en concreto en [42, 2]. En el primer caso, los elementos principales están explicados con mayor claridad. Además, los factores principales que influyen en la definición de la red viaria del transporte público. De hecho, las redes peatonal y la red viaria base están definidos en los capítulos 3 y 4 con precisión y simplicidad. Por lo tanto, no hay necesidad de repetición pero si de aclaración. Aunque no se haya tomado para resolver el modelo metropolitano completo de la ciudad de Sevilla, la red viaria expuesta en la figura 3.3 del mismo. La red peatonal del ámbito público puede entenderse definida dentro de las cualidades necesarias para el proyecto (se definen las vías de forma clara, especificando las zonas no transitables, las limitaciones que tiene el modelo a pie diseñado, remarcando las conexiones mas importantes que este modelo aporta) en la figura 4.16 necesitando únicamente la zona correspondiente a Sevilla.

Los factores clave que definen la red son:

- Número de carriles.
- Capacidad por carril.
- Velocidad en flujo libre.
- Congestión en los propios datos de *Google Maps*

El volumen de tráfico en cada posición queda indicado en esta clase de modelos utilizan como una matriz de flujo entre los pares origen destino (O-D). Por necesidad, se conoce el importante papel que las asignaciones tienen en la definición y evaluación del estado del transporte público. De esta sentencia, se pasaría al equilibrio definido por *Wardrop* en el capítulo 2 2.

Los resultados de la asignación se comentarán en los capítulos próximos pero se espera:

- El flujo siendo el número de vehículos por arco en un intervalo de tiempo.
- El tiempo de viaje calculado en cada red en el momento en que se realiza la asignación.
- El ratio volumen/capacidad siendo el cociente entre el numero de vehículos circulando en una arco por unidad de tiempo.

Se debe ser capaz de analizar como se desplaza la población en la capital en un día laboral cualquiera en los diferentes modos de transporte. Por la naturaleza de los datos, los factores comúnmente imprevistos como pueden ser obras en el modo a pie, desvíos por carretera, accidentes, retrasos por tráfico están incluidos dentro de los datos introducidos puesto que proceden de *Google Maps*.

Se debe recalcar la importancia que tienen las vías de uso exclusivo de Transporte público de autobuses en los tiempos de viaje. Reducirá los mismos junto a la congestión, evitando la influencia del tráfico privado. La apariencia de estas vías se puede apreciar en la figura 3.4.

Figura 3.3 Carril bus en Sevilla, calle José Laguillo *Tussam*.

Los ejemplos más llamativos del desatoro que producen se pueden en las zonas de tráfico más intenso en términos de congestión son los carriles únicos de la estación de buses Plaza de Armas o las vías del puente del Cachorro . Estos efectos junto a los de la red de vías bus que tiene la ciudad se tienen en cuenta dentro de los valores de tiempo que *Google Maps* responde a las peticiones.

Figura 3.4 Carril bus representado en amarillo sobre el puente del Cachorro. En la zona arriba a la derecha de la imagen se puede apreciar otro ejemplo de esta vía en la *estación Plaza de Armas*.

3.5 Resultados obtenidos

Se presentan los resultados obtenidos para la red de transporte definida principalmente de la siguiente forma:

- Conjunto de nodos compuesto por todas las paradas urbanas y centroides repartidos por toda Sevilla.
- Conjunto de *links* compuesto por: transbordos, mapa de viajero caminando y los enlaces del autobús urbano.

Esta información está compuesta a su vez por una lista que contiene 4 campos definidos en 3.1 3.2.

$$\textbf{link} = (\textbf{la}numbers\textbf{lo}numbers\_\textbf{la}numbers\textbf{lo}numbers, coste, frecuenciadelenlace, nombrerealdelaconexinentrenodos) \tag{3.1}$$

$$\textbf{link} = (\textbf{la}numbers\textbf{lo}numbers, coste, frecuencianodal, nombrerealdelnodo) \tag{3.2}$$

Considerando este esquema de organización de datos, se presentan una serie de mapas con el fin de ser ilustrativo, condensar el significado de la información obtenida y en la versión electrónica se incluye toda la información parada por parada y enlace por enlace.

La red a presente a continuación es la red de líneas de *Tussam* que opera en Sevilla posee una estructura radial con cada una de sus zonas unidas al centro con al menos una línea de autobús. Las líneas restantes (periféricas y transversales) permiten aumentar la cohesión de la red y diversificar el desplazamiento por cualquier zona de la ciudad.

La red de *Tussam* cuenta con 43 líneas diurnas y 9 nocturnas cuyo recorrido es de 660 km. El conjunto de vehículos es de 401 autobuses y 4 tranvías. En este proyecto se han tenido en cuenta la mayoría de líneas, no habiendo considerado la línea de tranvía, las líneas circulares ni las nocturnas. Siendo los números especificados algo menores a la magnitud expresada. El número de paradas es de 1012 con una distancia media de 300 metros. La frecuencia media de paso de los vehículos es de 8.75 cada hora con un intervalo entre vehículos de 13 minutos [33]. En la figura 3.5, se pueden apreciar el conjunto de carreteras de la red de transporte utilizada.

Se puede comprobar que esta red es muy similar a la proporcionada por Fernández et al. en la figura 4.1 del trabajo [42]. Las principales diferencias son la extensión (en este proyecto el área metropolitana no ha sido considerada) y la presentación de la simplificación que utiliza *Google Maps* para proporcionar la información sobre el recorrido que realiza el transporte público de la ciudad.

Seguidamente, se presentan en detalle los elementos que componen la red aparte de la red base de las carreteras que recorre para llevar a los viajeros a su destino.

3.5.1 Líneas

A continuación se presentan los mapas de las líneas de Sevilla. Previamente, se muestra donde quedan localizadas todas las paradas de autobús utilizadas en el mapa 3.6.

Transversales

Siendo las de mayor longitud conectan la ciudad con sus zonas mas alejadas y son:

- Línea 1: Polígono Norte - Virgen del Rocío
- Línea 2: Barqueta - Heliópolis
- Línea 3: Pino Montano - Bellavista
- Línea 5: P. Triana - Santa Aurelia
- Línea 6: San Lázaro - Virgen del Rocío

cuyos tramos de ida y vuelta se representan en la figura 3.7.

Cada una de ellas atraviesan Sevilla de norte a sur, exceptuando la 6, que lo hace de este a oeste.

Radiales Norte

Estas operan en la zona norte con una estructura radial que favorece el flujo de pasajeros hacia el centro de la ciudad y son:

- Línea 10: Ponce de León - San Jerónimo
- Línea 11: Ponce de León - Los Príncipes
- Línea 12: Ponce de León - Pino Montano
- Línea 13: Plaza del Duque - Pino Montano
- Línea 14: Plaza del Duque - Polígono Norte
- Línea 15: San Diego - Ponce de León
- Línea 16: Plaza San Jerónimo de Córdoba - Valdezorras

cuyos tramos de ida y vuelta se representan en la figura 3.9.

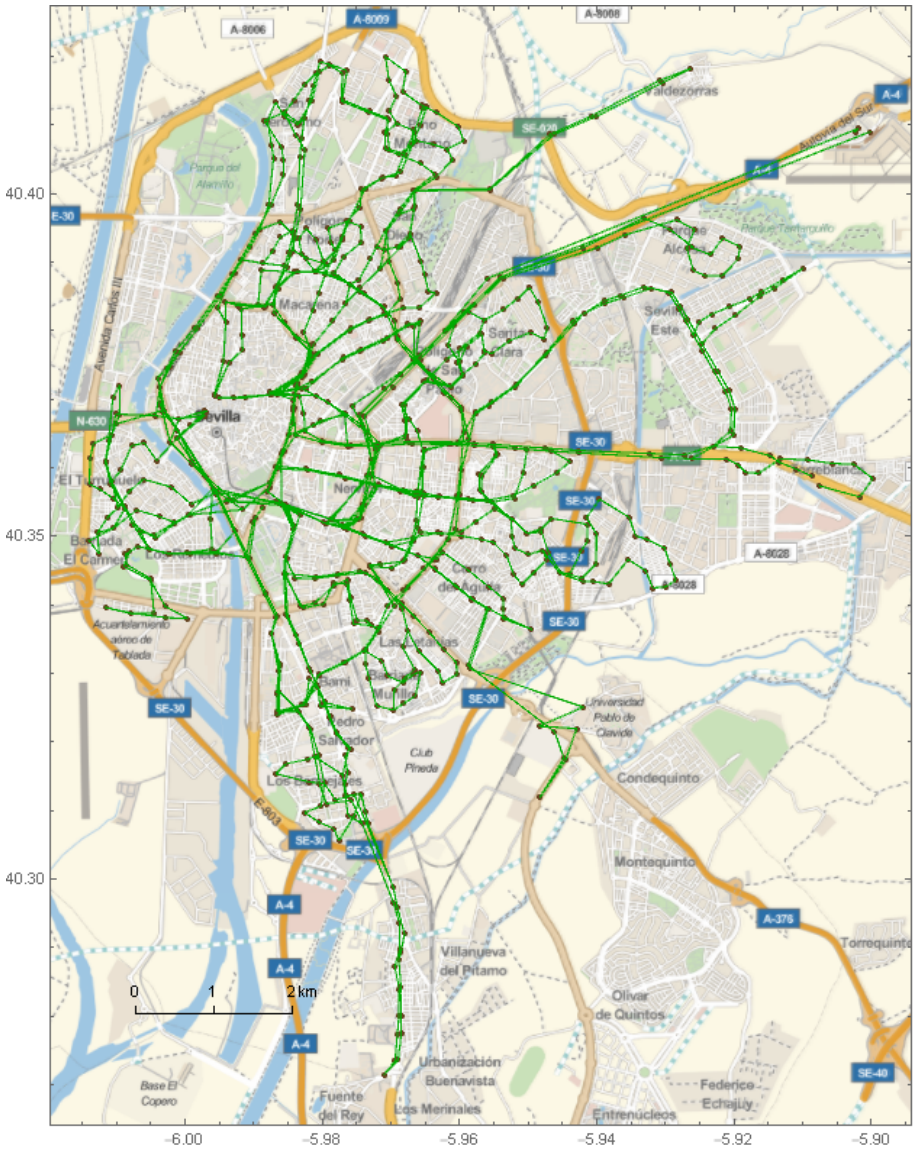

Figura 3.5 Red de autobuses *Tussam*.

Radiales Este

Funcionando de la misma forma que las anteriores, conectan los barrios periféricos de Sevilla con el centro de la ciudad y son:

- Línea 20: Ponce de León - Polígono San Pablo
- Línea 21: Plaza de Armas - Polígono San Pablo
- Línea 22: Prado San Sebastián - Sevilla Este
- Línea 24: Ponce de León - Palmete
- Línea 25: Rochelambert - Prado San Sebastián
- Línea 26: Prado San Sebastián - Cerro del Águila
- Línea 27: Plaza del Duque - Sevilla Este
- Línea 28: Parque Alcosa - Prado San Sebastián
- Línea 29: Prado San Sebastián - Torreblanca
- Línea 52: San Bernardo - Palmete

cuyos tramos de ida y vuelta se representan en la figura 3.9.

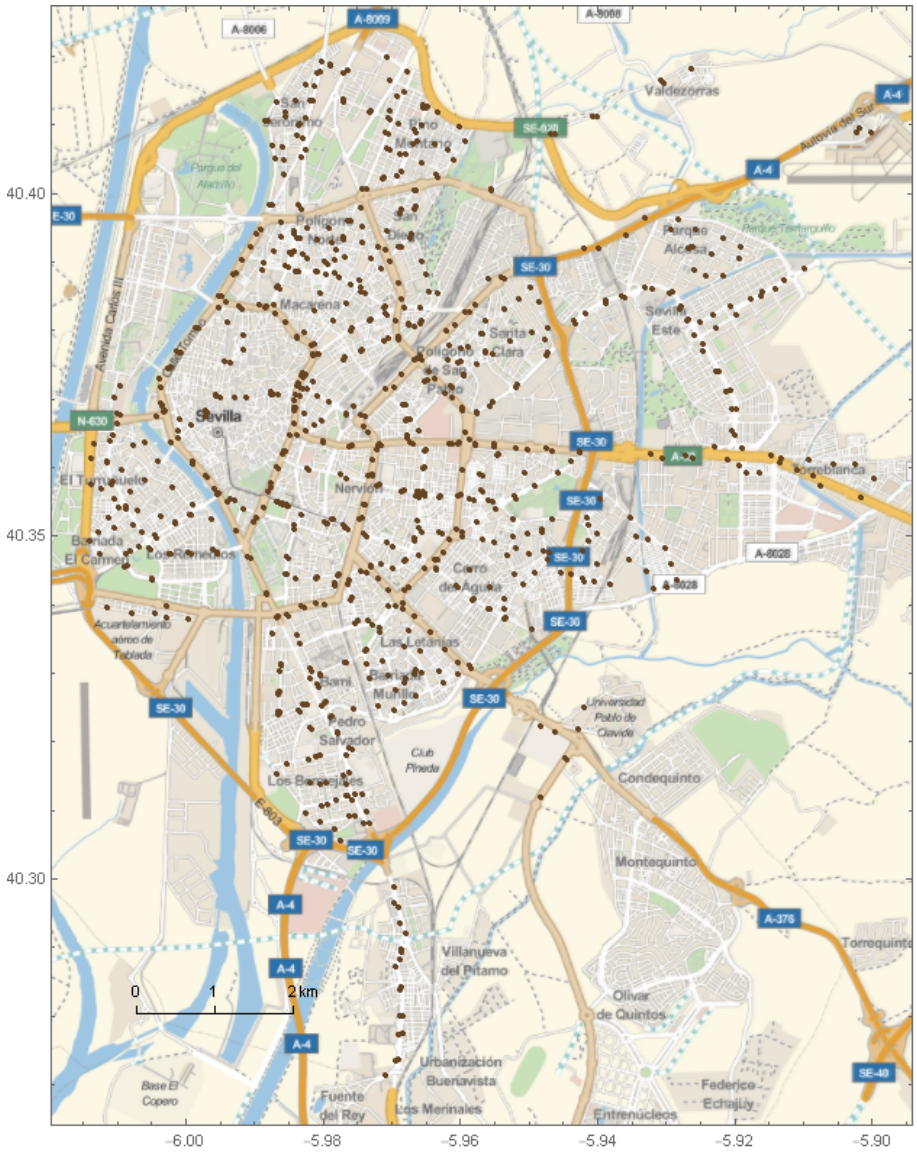

Figura 3.6 Paradas de la red *Tussam* creada con datos de *Google Maps*.

Radiales Sur

Su actividad se ubica en los distritos de Los Remedios, Palmera Bellavista y sur, nuevamente permitiendo el flujo hacia el centro de la ciudad y son:

- Línea 30: Prado San Sebastián - Barriada La Paz
- Línea 31: Polígono Sur - Prado San Sebastián
- Línea 32: Plaza del Duque - Polígono Sur
- Línea 34: Prado San Sebastián - Los Bermejales
- Línea 37: Puerta Jerez - Bellavista
- Línea 38A: Prado San Sebastián - Pítamo - U.P.O.

cuyos tramos de ida y vuelta se representan en la figura 3.10.

Radiales Oeste

Dan cobertura a la parte oeste, concretamente al distrito de Triana, uniendo este al centro de la ciudad y son:

- Línea 40: Plaza Magdalena - Triana

Figura 3.7 Líneas transversales sobre el mapa de la ciudad de Sevilla.

- Línea 41: Plaza Magdalena - Los Remedios - Tablada
- Línea 43: Plaza Magdalena - Triana - El Turruñuelo - El Tardón

cuyos tramos de ida y vuelta se representan en la figura 3.11.

Especiales

Se trata de las líneas que conectan el aeropuerto con Sevilla, junto a otras 2 que circundan y conectan los barrio como se puede apreciar en 3.12. Sus nombres:

- Línea B3: Santa Clara - Polígono San Pablo - Gran Plaza
- Línea EA: Plaza de Armas - Aeropuerto

3.5.2 Enlaces

Entre estos se pueden encontrar los siguientes:

- Transbordos
- Desde los centroides a las paradas.

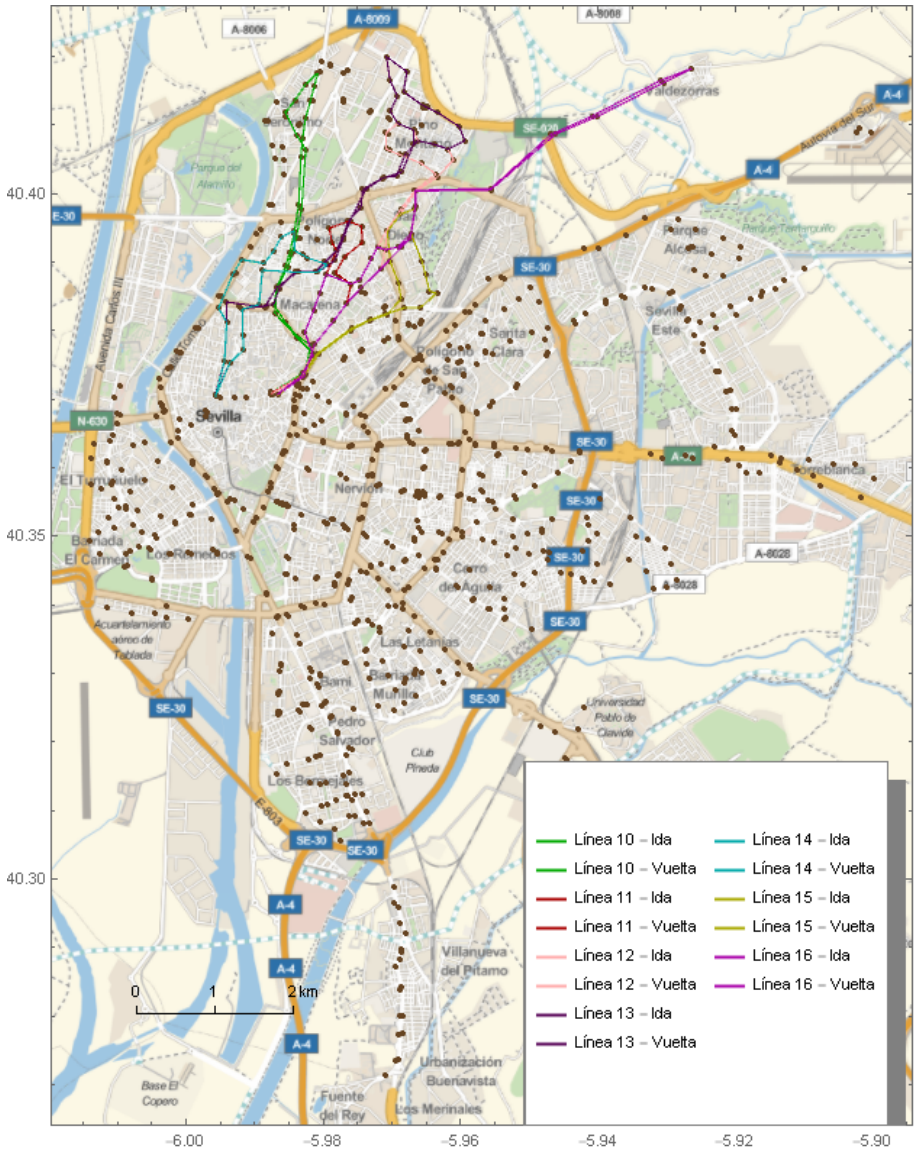

Figura 3.8 Líneas Radiales Norte sobre el mapa de la ciudad de Sevilla.

A continuación, se expondrán sus características, se resume su contenido para que sea percibido como conjunto de una forma conceptual fácil de asimilar debida al gran volumen de datos que se tienen y se explica su funcionamiento.

Desde los centroides a la red de autobuses

Zonas de transporte

Previo a la explicación de las zonas por las que se ha divido el distrito de la ciudad de Sevilla, se debe comentar y presentar las razones y la forma en la que se han distribuido.

La zonificación de Sevilla se obtiene de un estudio que el ayuntamiento realizó en el año 2007 sobre movilidad [43]. La zona de la ciudad de Sevilla se divide en 182 zonas de transporte sin contar con su área metropolitana (de las cuales se han utilizado 164 por motivos de líneas de autobús envueltas en el proyecto). La metodología para un estudio de demanda genérico consiste en estructurar a que zona le pertenece cada área de un modo lo mas homogéneo posible, comúnmente denominados *Zonas de Análisis de transporte* o TAZ.

Los viajes registrados en [43] de forma organizada dependiendo de la hora del viaje y las zonas O-D (origen destino) se almacenan en formato matricial T_{ij} con unas dimensiones nxn donde n es el número de centroides considerados. Esta matriz utilizada cuando se realiza la asignación en el capítulo de resultados fue proporcionada por el tutor del proyecto, considerando que su estimación era correcta, precisa cumpliendo

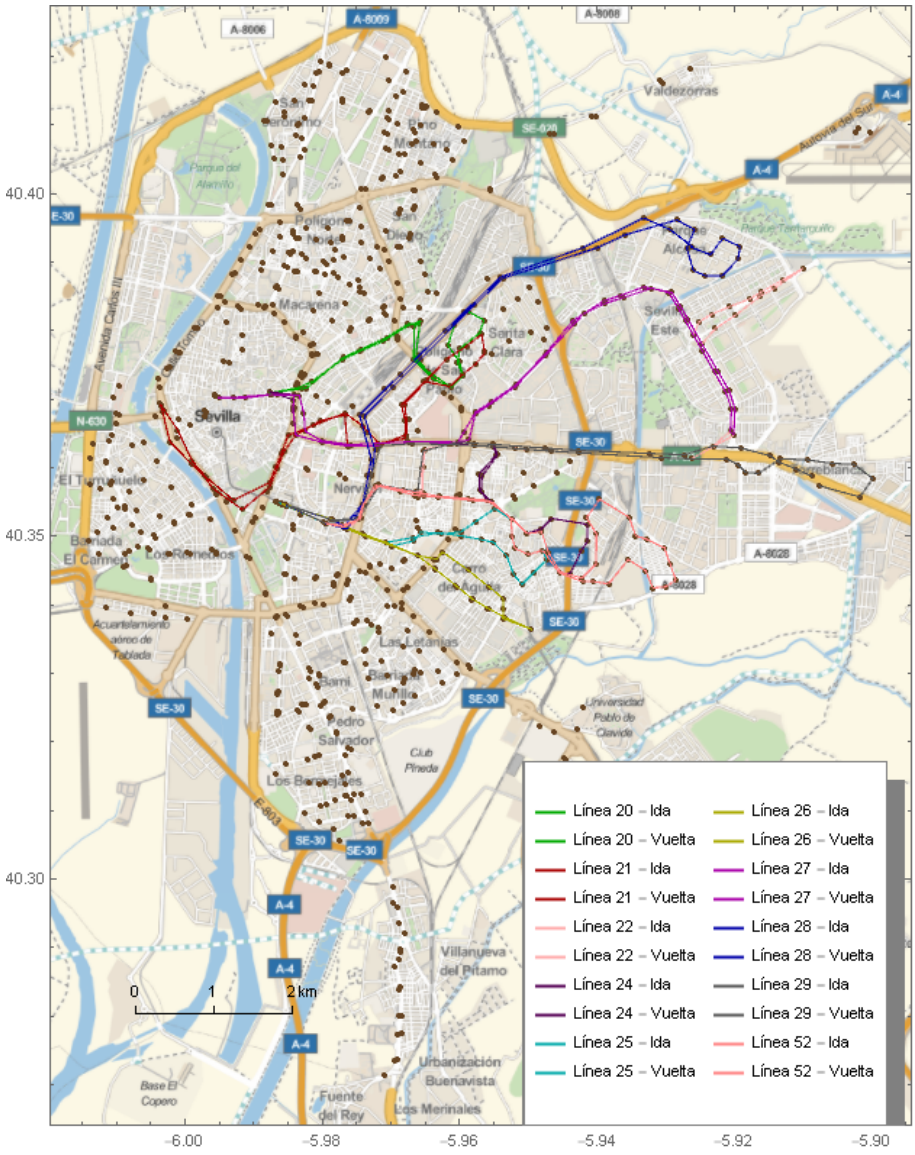

Figura 3.9 Líneas Radiales Este sobre el mapa de la ciudad de Sevilla.

lo necesario para que el cálculo realizado no resultase inútil. Se comentaba lo anterior atendiendo a que en algunas ocasiones esta aproximación de la misma resulta fundamental para comprender la situación del transporte público emn la ciudad.

Cada TAZ tiene un centroide que es el punto del cual se considera que parten o llegan los viajes que se relaciona con esa zona de transporte en análisis. Resulta obvio, que cuando más grande es el número de TAZ en una zona a estudiar, más pequeña en esta misma zona resultando en un detalle mejor con un carácter mas realista. Ahora, surge la necesidad de conectar el modelo a la red de transporte, es decir, unir mediante modos a pie los centroides con las paradas mas cercanas. Este paso se realizó considerando únicamente las paradas en un radio de 500 metros considerando el centroide como origen.

La creación de estos conectores viene motivada por la conexión necesaria del centroide a la red de transporte en esta macro simulación que concentra todos los viajes de una zona en los centroides.

Los factores más importantes para la creación de un TAZ son:

- Su compatibilidad con los distritos censales
- La compatibilidad del parcelario urbanístico de Sevilla.
- La compatibilidad con las vías urbanas del modelo.

Los TAZ utilizados y los centroides asociados quedan representados para todos Sevilla en 3.13.

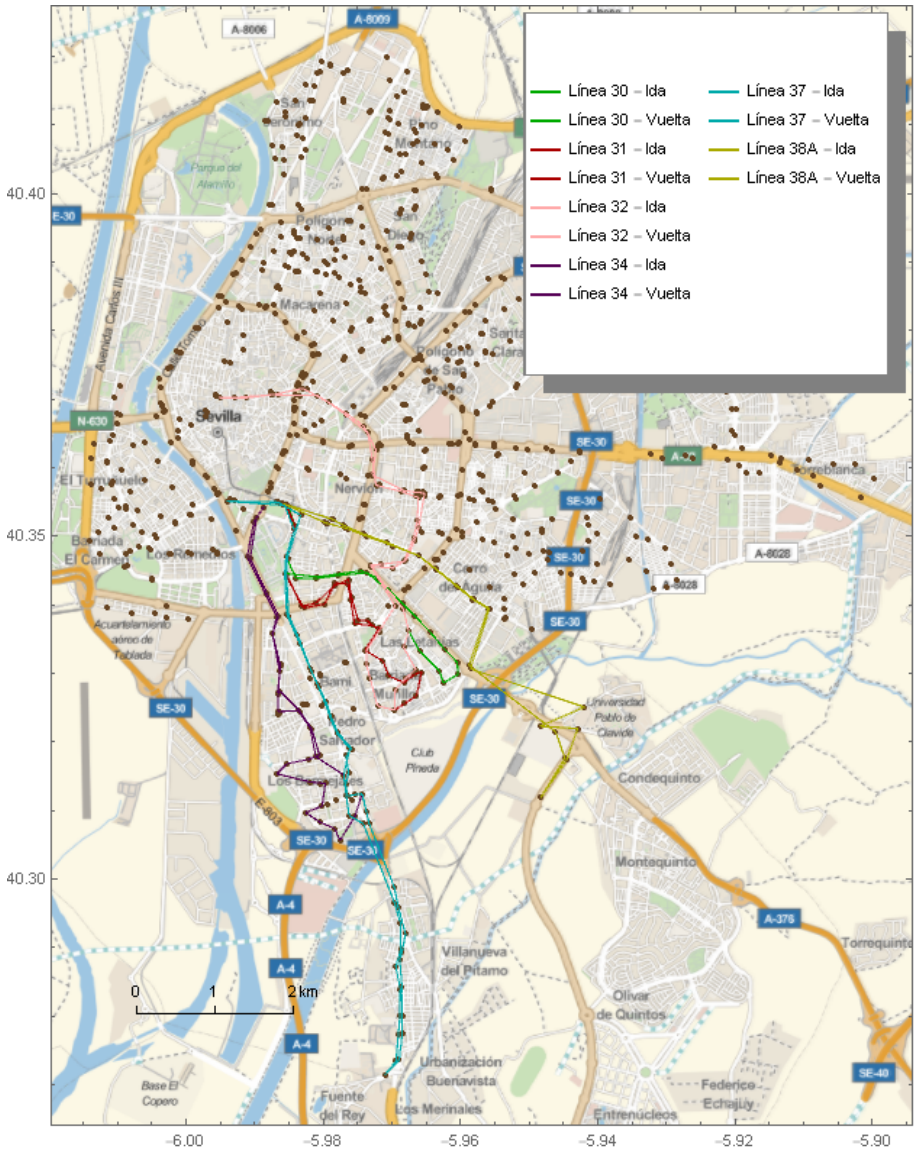

Figura 3.10 Líneas Radiales Sur sobre el mapa de la ciudad de Sevilla.

Para concluir esta subsección, habría que comentar la influencia de los conocidos como *carriles bus*. Aunque para explicar esta zona no hace falta detallarlos se deberá aportar información en el siguiente capítulo cuando se presente la asignación, puesto que esta en ocasiones utiliza vías de uso exclusivo mejorando la labor del transporte público y reduciendo el efecto de la congestión sobre este. Finalmente, cabe comentar que son utilizados como recurso para la consecución de los efectos previamente mencionados.

Faltaría comentar que aún habiendo mencionado los elementos de la red a pie, el efecto de los conectores de la misma y los conectores (entre centroides y sus zonas limítrofes de asignación de transporte) se pueden apreciar en 3.14. El caso del transbordo se puede apreciar en la siguiente subsección.

Matriz Origen Destino

En este caso, la matriz que fue proporcionada (se usaron 2: en hora punta y de diario) por el director del proyecto esta basada en [43] habiendo sido refinada después para adaptar los datos a esa gran diferencia de años que hay entre la fecha actual y la de la encuesta.

Aunque se utilizasen los datos de 2007, se debe considerar que la evolución del tráfico durante el intervalo de tiempo transcurrido. Mediante un estudio comparativo entre 2006 y 2016 de la intensidad del mismo, midiendo el número de vehículos por día en comparación con los datos publicados por la DGT, resaltan que ha habido un descenso progresivo de las intensidad de tráfico en los últimos años. Siendo sus máximos y

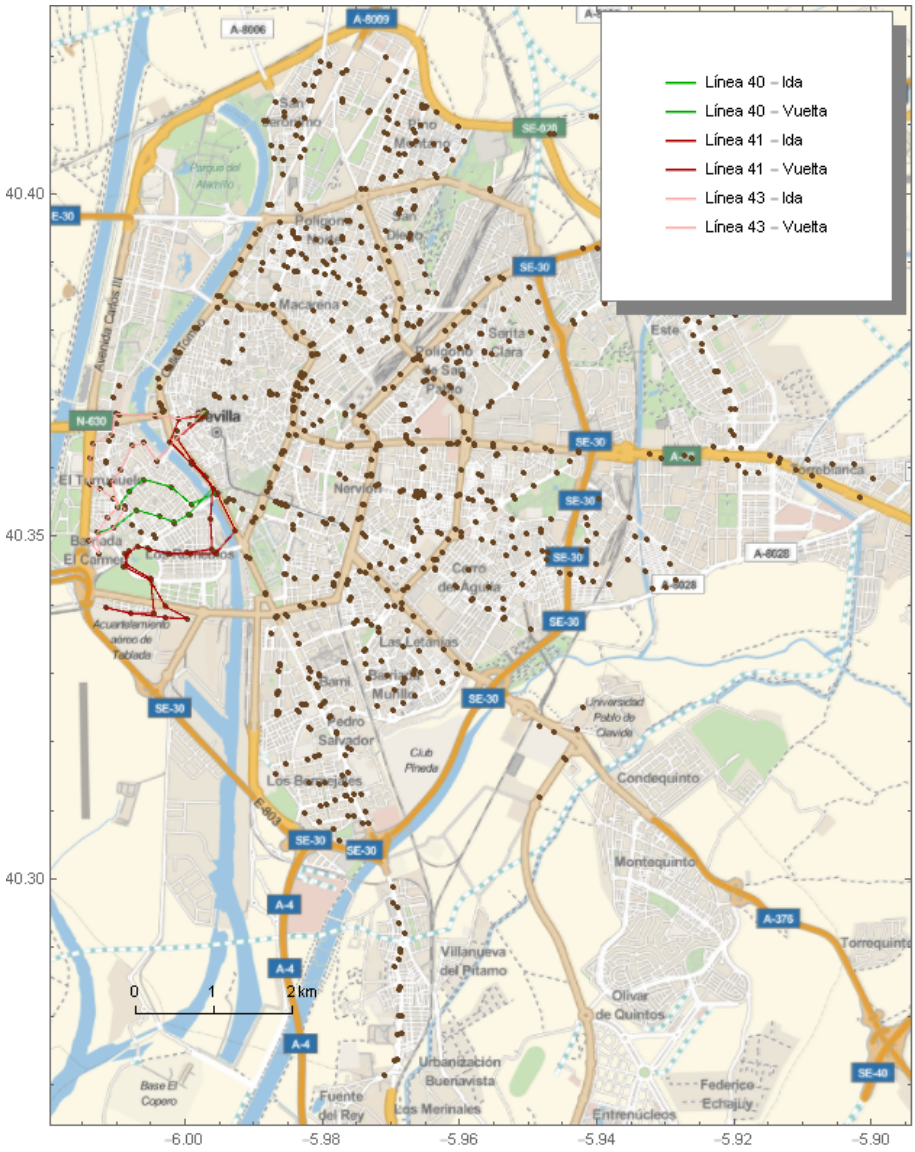

Figura 3.11 Líneas Radiales Oeste sobre el mapa de la ciudad de Sevilla.

mínimos, 2007 y 2015, considerando lo explicado anteriormente se llega a la conclusión de que el descenso es fruto de la crisis económica que asoló el país. Esta es una de sus repercusiones.

Por lo tanto, los datos que se extraen aún siendo conservadores resultan válidos para el cálculo en vista de la reducción de tráfico acontecida.

3.5.3 Conexión entre autobuses: Transbordos

Para terminar el capítulo, cabe mencionar la última componente del modo a pie, que se trata de los transbordos. La posibilidad de un viajero a cambiar de parada si la estrategia así lo contempla. Está penalizado su coste con un factor de 1.2 provocando un comportamiento del peatón más humano en lo que respecta a la toma de decisiones [2].

Quedaron definidos con una distancia para ser considerados de un radio de 200 metros desde cualquier parada. Se presentan varios ejemplos relacionados con una parada concreta en la figura 3.15. La red peatonal queda perfectamente presentada en la figura 4.16 del proyecto [42], aunque en este caso a escala mas reducida, considerando sólo la zona municipal de Sevilla. Se utilizan las mismas calles resaltadas en el documento para que el peatón conecte entre centroides y paradas, o entre transbordos. Dentro de los datos obtenidos mediante *Mathematica* y *Google Maps*, se considera el estado que tuvo la ciudad de Sevilla por completo (considerando obras, zonas cortadas y otro tipo de cambios) del día 16 de octubre, en hora punta y durante el

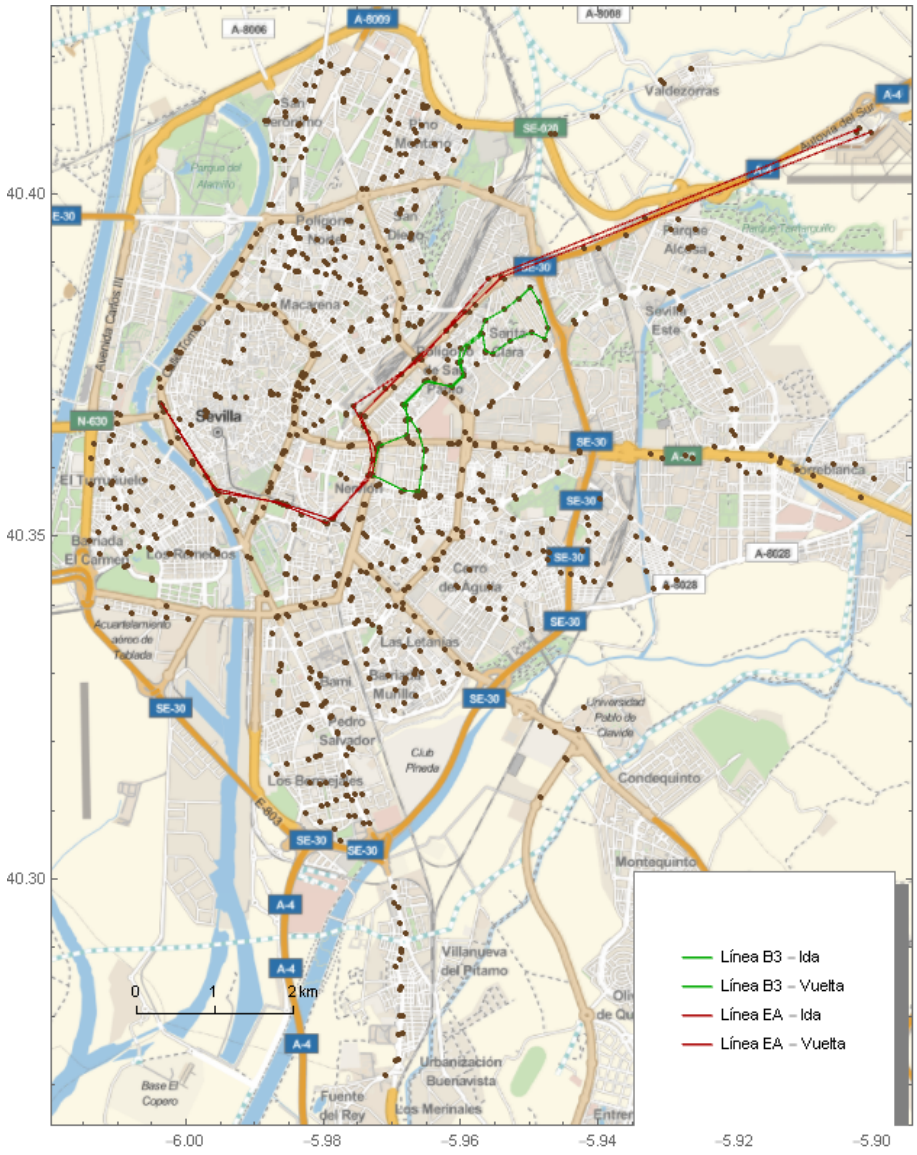

Figura 3.12 Líneas Especiales sobre el mapa de la ciudad de Sevilla.

día.

Cabe comentar que en algunos programas de asignación comercial como *TransCAD*, cuando se digitaliza la red de transporte y durante el proceso de conexión de la red viaria privada con el transporte público,

Finalmente, se debe explicar como se ha unido toda la base de datos. Mediante 2 ficheros excel que contienen por un lado, nodos y sus propiedades; por otro, todos los enlaces de esta red que unen cualquiera de esos nodos. Puesto que se mencionó anteriormente que la red no hace distinción entre andando o en autobús o haciendo un transbordo, lo contempla utilizando la frecuencia del *link*. Estos archivos son filtrados y tratados como se explica en el siguiente capítulo puesto el tratamiento de datos antes de usarlos en *Matlab* le corresponde.

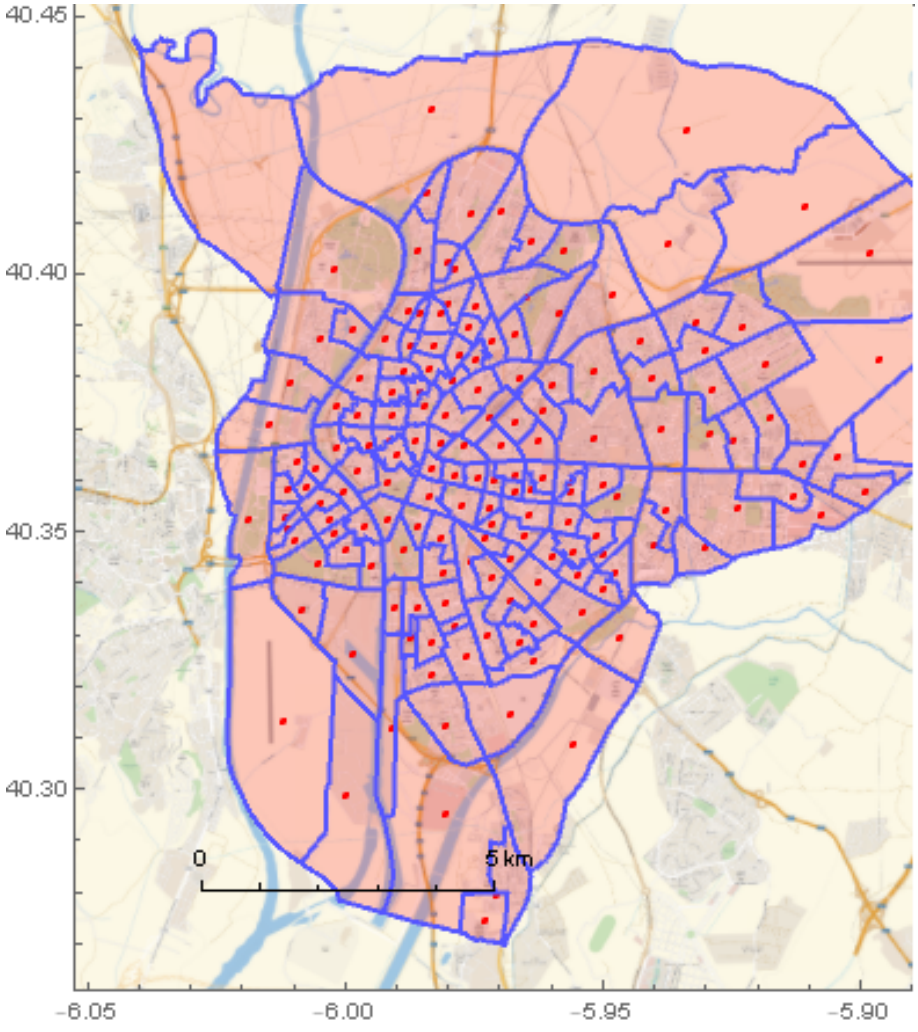

Figura 3.13 TAZ y sus centroides asociados en uso durante el proyecto.

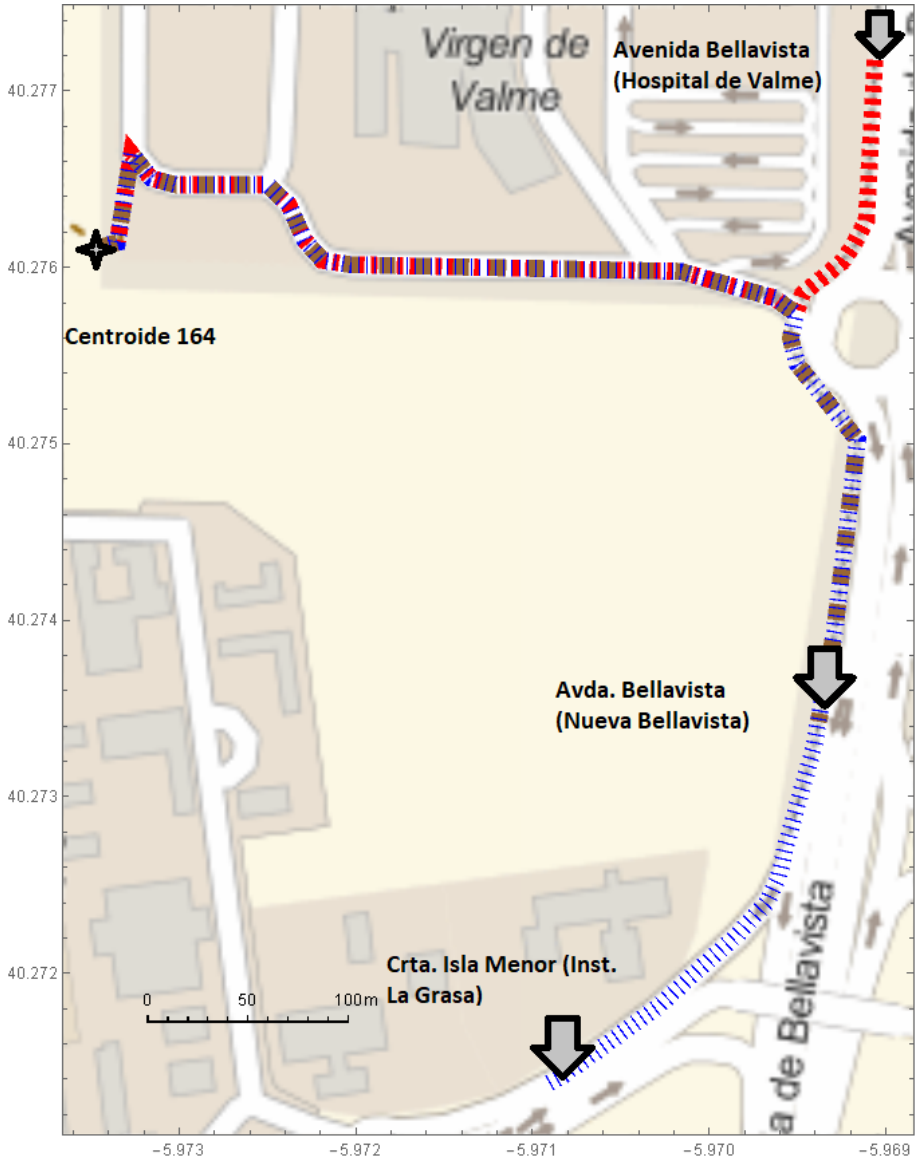

Figura 3.14 Conexión a pie entre los centroides y los márgenes de las TAZs. Como ejemplo se escogen las paradas mas cercanas al centroide 164: *Ctra. Isla Menor (Inst. La Grasa)*, *Avenida Bellavista (Hospital de Valme)* y *Avda. Bellavista (Nueva Bellavista)*.

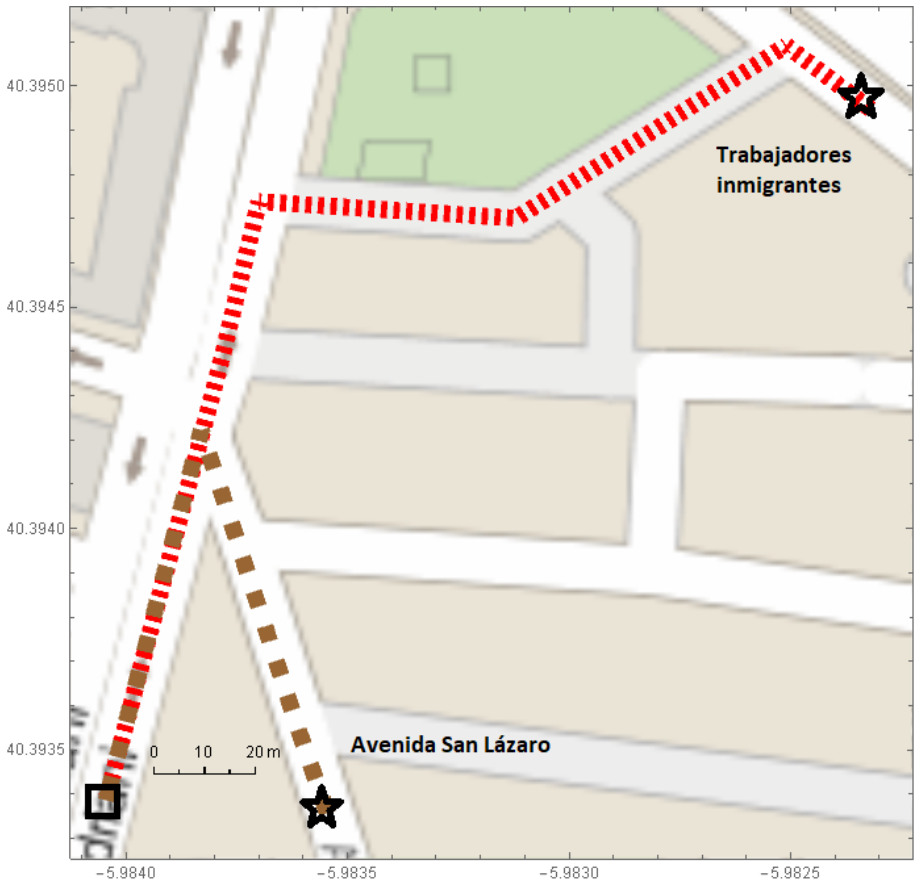

Figura 3.15 Transbordos posibles desde la parada de *Doctor Fedriani (Avda. San Lázaro)* hacia *Trabaj. Inmigrantes(Diego de Almagro)* y *Avenida San Lázaro (Doctor Morote)*. En el cuadrado sin rellenar de color negro se encuentra la primera parada mencionada..

4 Algoritmo de asignación de transporte público

Science is the belief in the ignorance of experts.

Richard Feynman

Aquí se explica detalladamente el funcionamiento del algoritmo descrito por *Spiess et Al.* [1] en 2 partes claramente diferenciadas. Primero, se determinan las estrategias óptimas de entre las posibles opciones disponibles en una ciudad respetando las premisas o hipótesis que hacen posible este modo de solución. Segundo, sabiendo el volumen de personas que haya en cada parada, se resuelve el problema asignando el reparto de pasajeros por las estrategias óptimas en orden topológico inverso al de las rutas óptimas.

Se debe recordar que mientras mas información reciba el viajero se hace mas difícil la decisión a tomar junto al algoritmo que describiría esta solución óptima. Un ejemplo de esta información adicional que no se considera en este proyecto podría ser: información geolocalizada y de tiempo restante sobre otros vehículos que circulan cuando se está circulando en otro vehículo, tiempos de llegada al esperar en una parada, entre otros. Por lo tanto, para definir el algoritmo las hipótesis de *Spiess* son las siguientes:

- La única información disponible para el viajero es la siguiente línea de autobús que pasará en la parada donde se encuentran esperando.
- No se consideran estrategias circulares o que tienen el mismo sitio de partida y llegada.
- Se llamará a cada elemento escogido por el viajero *strategy*. Sin diferenciar si es un nodo (parada), un link (viaje en vehículo desde un nodo a otro), viaje andado (walking mode), entre otros ya descritos previamente.

Tras las asunciones, resulta importante las diferentes componentes de viaje que se definen en el modelo del artículo:

- Desde el origen hasta la parada del transporte
- Espera a un vehículo
- Montado en el vehículo
- Bajándose del vehículo
- Enlace andando entre 2 paradas
- Salida desde la parada de transporte hasta el destino

Por consiguiente, se define el modelo de transporte público como una red conectada de nodos y enlaces entre nodos donde intervienen dependiendo de el objeto las componentes de diferente forma. *Spiess* en su artículo ejemplifica la trasformación de un esquema de transporte conceptual (fig.1 [1]) a la red de transporte público que describe en la figura 5 del mismo. Este modelado ha sido aplicado a la ciudad de Sevilla obteniendo la información necesaria desde diferentes fuentes como *Google Maps* y la página web de Tussam, empresa que gestiona el transporte público de la misma.

4.1 Asignación descrita por Spiess

El algoritmo planteado por Spiess se presenta de la forma 1.

Algorithm 1 Versión original de Spiess

procedure $Part1 : Find \quad optimal \quad strategy$
  $u_i := \infty, i \in I - r; u_r := 0;$ ▷ 1.1 initialization
  $f_i := 0, i \in I;$
  $S := A; \bar{A} := \emptyset$
  while $S \neq \emptyset$ **do** ▷ 1.2 Get the next link
    $find \quad a = (i,j) \in \quad S \quad which \quad satisfies$
    $u_j + c_a \leq u_{j'} + c_{a'}, \quad a' = (i',j') \in S;$
    $S := S - a$
    if $S := 0$ **then**
      $STOP$
    end if
    if $u_i \geq u_j + c_a$ **then** ▷ 1.3 Update node label
      $u_i := \frac{f_i u_i + f_a(u_j + c_a)}{f_i + f_a}$
      $f_i := f_i + f_a, \quad \bar{A} := \bar{A} + a;$
    end if
  end while
end procedure
procedure $Part2 : Assign \quad demand \quad according \quad to \quad optimal \quad strategy$
  $V_i := g_i; \quad i \in I;$ ▷ 2.1 initialization
  for $every \quad a \in A \quad in \quad decreasing \quad order \quad (u_j + c_a)$ **do** ▷ 2.2 Loading
    if $a \in \bar{A}$ **then**
      $v_a := \frac{f_a}{f_i} V_i,$
      $V_j := V_j + v_a$
    else
      $v_a := 0$
    end if
  end for
end procedure

Consecuentemente, el algoritmo descrito en [1] se expresa en *Matlab* cuyo resultado y resolución quedan descritas en 4.1. Este código lleva dentro los datos de la red propuesta por *Spiess* aplicados a la Fig. 6 del citado artículo. Al ejecutarlo se obtiene la solución descrita en [1]. Este algoritmo es la versión que incluye los datos introducidos en el propio código.

Código 4.1 Algoritmo de Spiess - Matlab Version.

```
%% Part 1 Find optimal strategy

% 1.0 filling the arrays (this part is prepared for spiess and ) Map
% structure try
nodes = struct('A',{},... % that is spiess data structure. They must contain (
    waiting from i to j, and frec)
              'X2',{},...
              'X0',{},...
              'Y3',{},...
              'Y0',{},...
              'B',{});

% 1.1 Initialization. Assigment of previous values of Spiess and Florian
%u_r = 0; % u_i = infinite; Para todo i en I menos r (destination node)
```

```
%f_i = 0; % para todo i en I

Rnod = 'B';  % destination node
fild = fieldnames(nodes);
for i = 1:length(fild)
    if fild{i}==Rnod
        nodes(1).(sprintf('%s',fild{i})) = 0;% using sprintf we can access to
            the data using the name
        nodes(2).(sprintf('%s',fild{i})) = 0;
    else
        nodes(1).(sprintf('%s',fild{i})) = inf;
        nodes(2).(sprintf('%s',fild{i})) = 0;
    end
end

% links: they follow a basic rule of left to right lecture WHICH also indicates
% the direction of the node. Previous simplification of the data is need to
% achieve proper results because of the complexity of data introduced and
% the nodal relation. label, {cost, frecuency}
%S = A; % Not(A) = '';
links = struct('A_X2',{7,1/6},...
               'A_B',{25,1/6},...
               'X2_X0',{0,inf},...
               'X0_X2',{0,1/6},...
               'X2_Y0',{6,inf},...
               'X0_Y3',{4,1/15},...
               'Y3_Y0',{0,inf},...
               'Y0_Y3',{0,1/15},...
               'Y3_B',{4,inf},...
               'Y0_B',{10,1/3});
% removing field links = rmfield(links,'A_B')

S = links;
Abar ={};

% 1.2 Get the next link
g = 1;
% used_nodes{1} = Rnod;
% un = 2;
iter = 1;
reverse_ord = {};
while not(isempty(fieldnames(S)))
    % starting the algorithm
    %    a = (i,j); % perte a S
    % find a which satisfies
    %    u_j + c_a <= u_jpr + c_apr; apr = (ipr,jpr);
    %    S = S - a;
    %    % 1.3 Update node label (Falta un trozo)
    %    ui = (fi*ui + fa*(uj+ca))/(fi+fa);
    %    fi = fi + fa;
    %    Abar = Abar + a;
    tips = fieldnames(nodes);
    linkes = fieldnames(S);
    j = 1;
    % getting links involved
    for i = 1:length(linkes)
       %    u_j + c_a
```

```
       Lparts = strsplit(char(linkes(i)),'_');
       posi = find(~cellfun(@isempty,strfind(tips,Lparts{2})));
       uj = nodes(1).(sprintf('%s',tips{posi}));
       ca = S(1).(sprintf('%s',linkes{i}));
       mat_uj_ca(j,:) = [uj+ca ...
           S(2).(sprintf('%s',linkes{i})) i ];
       j = j+1;
    end
    [m,indss] = min(mat_uj_ca(:,1));
    % getting node time info
    Lparts = strsplit(char(linkes(mat_uj_ca(indss,3))),'_');
    posi = find(~cellfun(@isempty,strfind(tips,Lparts{1})));
    ui = nodes(1).(sprintf('%s',tips{posi}));
    fi = nodes(2).(sprintf('%s',tips{posi}));
    reverse_ord{iter} = linkes(mat_uj_ca(indss,3));
    % eliminating link
    S = rmfield(S,linkes(mat_uj_ca(indss,3)));

    disp('iteration')
    struct2table(nodes)
    % checking if it belongs to the optimal strategy
    if ui >=( mat_uj_ca(indss,1))
        % cheking if infinite value is involved
        fa = mat_uj_ca(indss,2);
        if isinf(fi) | g==1 | (ui == inf && fi == 0 && isinf(fa))
           nodes(1).(sprintf('%s',tips{posi})) = mat_uj_ca(indss,1);
           nodes(2).(sprintf('%s',tips{posi})) = inf;
        elseif ui == inf && fi == 0
            nodes(1).(sprintf('%s',tips{posi})) = (1 + fa*(mat_uj_ca(indss,1)))
                /(fi+fa);
            nodes(2).(sprintf('%s',tips{posi})) = fa + fi;
        else
            nodes(1).(sprintf('%s',tips{posi})) = (ui*fi + fa*(mat_uj_ca(indss
                ,1)))/(fi+fa);
            nodes(2).(sprintf('%s',tips{posi})) = fa + fi;
        end
        Abar{g} = linkes(mat_uj_ca(indss,3));
        g = g+1;
    end
    % updating

    clear mat_uj_ca
    iter = iter +1;
    % remeber if
end
struct2table(nodes)
disp('Optimal strategies found')

cell2table(Abar) % links related to optimal strategies

%% Part 2 Assign demand according to optimal strategy

% 2.1 Initialization
%V_i = g_i; % para todo i en I
% reverse_ord must be used in inverse order
```

```
Vi = struct('A',{1},... % that is spiess data structure. They must contain (
    waiting from i to j, and frec)
              'X2',{0},...
              'X0',{0},...
              'Y3',{0},...
              'Y0',{0},...
              'B',{-1});

va = struct('A_X2',{0},...
          'A_B',{0},...
          'X2_X0',{0},...
          'X0_X2',{0},...
          'X2_Y0',{0},...
          'X0_Y3',{0},...
          'Y3_Y0',{0},...
          'Y0_Y3',{0},...
          'Y3_B',{0},...
          'Y0_B',{0});


% 2.2 Loading
struct2table(va)
struct2table(Vi')
k = 0;
for i = length(reverse_ord):-1:1 % in decreasing order of (u_i + c_a)
    for j = 1:length(Abar)
        if strcmp(Abar{j},reverse_ord{i})
            Lparts = strsplit(char(reverse_ord{i}),'_');
            % fi
            fi = nodes(2).(sprintf('%s',Lparts{1}));
            % fa
            fa = links(2).(sprintf('%s',char(reverse_ord{i})));
            if isinf(fa) && isinf(fi)
                Lparts = strsplit(char(reverse_ord{i}),'_');
                % fi and % fa are inf
                va.(sprintf('%s',char(reverse_ord{i}))) = Vi.(sprintf('%s',
                    Lparts{1}));
                Vi.(sprintf('%s',Lparts{2})) = Vi.(sprintf('%s',Lparts{2})) +...
                va.(sprintf('%s',char(reverse_ord{i})));
            else
                va.(sprintf('%s',char(reverse_ord{i}))) = Vi.(sprintf('%s',
                    Lparts{1}))...
                *fa/fi;
                Vi.(sprintf('%s',Lparts{2})) = Vi.(sprintf('%s',Lparts{2})) +...
                va.(sprintf('%s',char(reverse_ord{i})));
            end

        end
    end
    struct2table(va)
    struct2table(Vi')


end
```

En 4.1 las principales variables utilizadas fueron:

- Frecuencia nodo f_i

- Frecuencia enlace f_a
- El nodo es el subíndice i
- El enlace es el subíndice a y es descrito como $a = (i, j)$
- g_i es la demanda del nodo
- V_i es el volumen del nodo
- v_a es volumen de un enlace

Es importante remarcar la filosofía de resolución del algoritmo utilizado en varios sencillos puntos:

- Utilizar tipo *struct* facilitando la labor de programador mediante el uso de etiquetas asociadas de los enlaces y nodos con formato tipo *char* asociados a su longitud y latitud como se vio en el capítulo 3. Para consultas sobre el tipo de datos [19].
- Posteriormente, se cambiará la filosofía que funcionó de forma tan óptima con un volumen reducido de datos debido a la lentitud asociada al tipo utilizado. Se implementó de nuevo en tipo *Cell* consiguiendo reducir el cálculo de un mapa de rutas óptimas a 20 minutos desde 45 minutos.
- Esta versión utilizada cuenta con la solución que propone *Spiess* cuando describe su problema mediante infinitos. Adicionalmente, se comprobó que la solución de sustituir los infinitos por números muy grandes funciona correctamente como era previsible. [1]

Los resultados obtenidos para este caso son idénticos a los presentados por *Spiess y Florian* en su artículo.

4.2 Asignación con el ejemplo de Spiess sin simplificar

El código 4.1 se mantiene inalterado. Sin embargo, los arrays *nodes* y *links* cambian para incluir el ejemplo de la figura 5 recogido en el artículo [1]. El código resultante con los cambios pertinentes es descrito en 4.2.

Código 4.2 Algoritmo de Spiess - Matlab Version - Basic Network.

```
%% Part 1 Find optimal strategy

%% Optimal strategies by Spiess and florian (main script)
% This software was implemented by Javier Vázquez Peralta
% TFM Máster en diseño avanzado en ingeniería mecánica
% Supervisor: Luis Miguel Romero Pérez
%
%
%

clc;
clear all;

%% Part 1 Find optimal strategy

% 1.0 filling the arrays (this part is prepared for spiess and ) Map
% structure try
nodes = struct('A1',{},... % that is spiess data structure. They must contain (
    waiting from i to j, and frec)
            'A2',{},...
            'A0',{},...
            'X2',{},...
            'X3',{},...
            'X0',{},...
            'Y3',{},...
            'Y4',{},...
```

```
              'Y2',{},...
              'Y0',{},...
              'B2',{},...
              'B3',{},...
              'B4',{},...
              'B0',{});

% 1.1 Initialization. Assigment of previous values of Spiess and Florian
%u_r = 0; % u_i = infinite; Para todo i en I menos r (destination node)
%f_i = 0; % para todo i en I

Rnod = 'B0';  % destination node
fild = fieldnames(nodes);
for i = 1:length(fild)
    if fild{i}==Rnod
        nodes(1).(sprintf('%s',fild{i})) = 0;% using sprintf we can access to
            the data using the name
        nodes(2).(sprintf('%s',fild{i})) = 0;
    else
        nodes(1).(sprintf('%s',fild{i})) = inf;
        nodes(2).(sprintf('%s',fild{i})) = 0;
    end
end

% links: they follow a basic rule of left to right lecture WHICH also indicates
% the direction of the node. Previous simplification of the data is need to
% achieve proper results because of the complexity of data introduced and
% the nodal relation. label, {cost, frecuency}
%S = A; % Not(A) = '';
links = struct('A0_A2',{0,1/6},...
              'A0_A1',{0,1/6},...
              'A1_B2',{25,inf},...
              'A2_X2',{7,inf},...
              'B2_B0',{0,inf},...
              'X2_X0',{0,inf},...
              'X0_X2',{0,1/6},...
              'X2_Y2',{6,inf},...
              'Y2_Y0',{0,inf},...
              'X0_X3',{0,1/15},...
              'X3_Y3',{4,inf},...
              'Y3_Y0',{0,inf},...
              'Y0_Y3',{0,1/15},...
              'Y0_Y4',{0,1/3},...
              'Y3_B3',{4,inf},...
              'B3_B0',{0,inf},...
              'Y4_B4',{10,inf},...
              'B4_B0',{0,1/3});
% removing field links = rmfield(links,'A_B')

S = links;
Abar ={};

% 1.2 Get the next link
g = 1;
% used_nodes{1} = Rnod;
% un = 2;
iter = 1;
```

```
reverse_ord = {};
while not(isempty(fieldnames(S)))
    % starting the algorithm
    %    a = (i,j); % perte a S
    % find a which satisfies
    %    u_j + c_a <= u_jpr + c_apr; apr = (ipr,jpr);
    %    S = S - a;
    %    % 1.3 Update node label (Falta un trozo)
    %    ui = (fi*ui + fa*(uj+ca))/(fi+fa);
    %    fi = fi + fa;
    %    Abar = Abar + a;
    tips = fieldnames(nodes);
    linkes = fieldnames(S);
    j = 1;
    % getting links involved
    for i = 1:length(linkes)
       %    u_j + c_a
       Lparts = strsplit(char(linkes(i)),'_');
       posi = find(~cellfun(@isempty,strfind(tips,Lparts{2})));
       uj = nodes(1).(sprintf('%s',tips{posi}));
       ca = S(1).(sprintf('%s',linkes{i}));
       mat_uj_ca(j,:) = [uj+ca ...
           S(2).(sprintf('%s',linkes{i})) i ];
       j = j+1;
    end
    [m,indss] = min(mat_uj_ca(:,1));
    % getting node time info
    Lparts = strsplit(char(linkes(mat_uj_ca(indss,3))),'_');
    posi = find(~cellfun(@isempty,strfind(tips,Lparts{1})));
    ui = nodes(1).(sprintf('%s',tips{posi}));
    fi = nodes(2).(sprintf('%s',tips{posi}));
    reverse_ord{iter} = linkes(mat_uj_ca(indss,3));
    % eliminating link
    S = rmfield(S,linkes(mat_uj_ca(indss,3)));

    disp('iteration')
    struct2table(nodes)
    % checking if it belongs to the optimal strategy
    if ui >=( mat_uj_ca(indss,1))
        % cheking if infinite value is involved
        fa =  mat_uj_ca(indss,2);
        if isinf(fi) | g==1 | (ui == inf && fi == 0 && isinf(fa))
           nodes(1).(sprintf('%s',tips{posi})) = mat_uj_ca(indss,1);
           nodes(2).(sprintf('%s',tips{posi})) = inf;
        elseif ui == inf && fi == 0
            nodes(1).(sprintf('%s',tips{posi})) = (1 + fa*(mat_uj_ca(indss,1)))
                /(fi+fa);
            nodes(2).(sprintf('%s',tips{posi})) = fa + fi;
        else
            nodes(1).(sprintf('%s',tips{posi})) = (ui*fi + fa*(mat_uj_ca(indss
                ,1)))/(fi+fa);
            nodes(2).(sprintf('%s',tips{posi})) = fa + fi;
        end
        Abar{g} = linkes(mat_uj_ca(indss,3));
        g = g+1;
    end
    % updating
```

```
    clear mat_uj_ca
    iter = iter +1;
    % remeber if
end
struct2table(nodes)

disp('Optimal strategies found')

cell2table(Abar) % links related to optimal strategies


%% Part 2 Assign demand according to optimal strategy

% 2.1 Initialization
%V_i = g_i; % para todo i en I
% reverse_ord must be used in inverse order

Vi = struct('A1',{0},... % that is spiess data structure. They must contain (
    waiting from i to j, and frec)
              'A2',{0},...
              'A0',{1},...
              'X2',{0},...
              'X3',{0},...
              'X0',{0},...
              'Y3',{0},...
              'Y4',{0},...
              'Y2',{0},...
              'Y0',{0},...
              'B2',{0},...
              'B3',{0},...
              'B4',{0},...
              'B0',{-1});

va = struct('A0_A2',{0},...
              'A0_A1',{0},...
              'A1_B2',{0},...
              'A2_X2',{0},...
              'B2_B0',{0},...
              'X2_X0',{0},...
              'X0_X2',{0},...
              'X2_Y2',{0},...
              'Y2_Y0',{0},...
              'X0_X3',{0},...
              'X3_Y3',{0},...
              'Y3_Y0',{0},...
              'Y0_Y3',{0},...
              'Y0_Y4',{0},...
              'Y3_B3',{0},...
              'B3_B0',{0},...
              'Y4_B4',{0},...
              'B4_B0',{0});


% 2.2 Loading
struct2table(va)
struct2table(Vi)
```

```
k = 0;
for i = length(reverse_ord):-1:1 % in decreasing order of (u_i + c_a)
    for j = 1:length(Abar)
        if strcmp(Abar{j},reverse_ord{i})
            Lparts = strsplit(char(reverse_ord{i}),'_');
            % fi
            fi = nodes(2).(sprintf('%s',Lparts{1}));
            % fa
            fa = links(2).(sprintf('%s',char(reverse_ord{i})));
            if isinf(fa) && isinf(fi)
                Lparts = strsplit(char(reverse_ord{i}),'_');
                % fi and % fa are inf
                va.(sprintf('%s',char(reverse_ord{i}))) = Vi.(sprintf('%s',
                    Lparts{1}));
                Vi.(sprintf('%s',Lparts{2})) = Vi.(sprintf('%s',Lparts{2})) +...
                va.(sprintf('%s',char(reverse_ord{i})));
            else
                va.(sprintf('%s',char(reverse_ord{i}))) = Vi.(sprintf('%s',
                    Lparts{1}))...
                *fa/fi;
                Vi.(sprintf('%s',Lparts{2})) = Vi.(sprintf('%s',Lparts{2})) +...
                va.(sprintf('%s',char(reverse_ord{i})));
            end

        end
    end
    struct2table(va)
    struct2table(Vi)


end
```

Los resultados son igualmente perfectos, pero cambian la forma de interpretarse puesto que el grafo con el que se describen es diferente.

A continuación, se ha implementado la solución considerando los siguientes problemas:

- El tiempo de ejecución de cada cálculo puesto que se tienen que resolver 164 rutas óptimas junto a las asignaciones.
- La forma de interaccionar con los datos con respecto del anterior las secciones anteriores ha cambiado, no hay introducción de datos leída directamente en el script, se accede a las bases de datos externas que contienen la información del nodo, su frecuencia y su costo, e idénticamente para los enlaces.
- Se incorpora una función de guardado de datos de rutas óptimas para garantizar al usuario que no perderá información en caso de olvidar guardar.
- El que se ha comprobado previamente, que se pueden obtener los resultados incluso cuando no se llegue a la malla más óptima posible.
- El volumen de datos hace imposible presentar un gráfico que se pueda apreciar por alguien que no sea un experto.
- Se debe comentar que la base de datos puede dar algún problema menor y localizado debido a que durante la obtención de datos en *Google Maps* muchos de ellos experimentaron una variación en la millonésima o diezmillonésima. Como la selección de rutas óptimas se hace por comparativa de de tipo *String* es un factor importante a considerar.

Por lo tanto, se presenta el algoritmo final con el que se obtienen las asignaciones para la ciudad de Sevilla.

4.3 Código final utilizado para resolver la ciudad de Sevilla

Como fue descrito en el párrafo anterior, se resuelven las rutas óptimas para la ciudad de Sevilla guardando para cada destino (cada centroide) estas variables:

- La información de los nodos
- La información de los enlaces
- La información de los enlaces que componen las rutas óptimas.
- La ruta inversa de la óptima para aplicarla en la asignación.

Se consideraron de antemano, los problemas que el modelo del algoritmo mediante infinitos podría ocasionar en el caso de que no se programe adecuadamente. Sin embargo, como se muestra en la sección siguiente y mediante las sugerencia de Spiess et. al. en [1] sobre la gestión de los mismos, y una buena depuración del código de forma pormenorizada, se solventaron los problemas de datos tipo *NaN* o errores durante la ejecución del código mismo [19].

Se resolvió utilizando *Matlab* un problema de unos 1100 nodos y 10000 enlaces de la modalidad que fuese. Cada cálculo se resolvía en unos 25 minutos aproximadamente. El código encargado de realizar todos los cálculos para las rutas óptimas se puede consultar en 4.3.

Código 4.3 Algoritmo de Spiess - Matlab Version - Red de Sevilla.

```
%% Optimal strategies by Spiess and florian (main script)
% This software was implemented by Javier Vázquez Peralta
% TFM Máster en diseño avanzado en ingeniería mecánica
% Supervisor: Luis Miguel Romero Pérez
%

%% 1st test with Sevilla city information
%
%  The data obtained after filtering the raw infromation can be obtained in
%  order to solve the assigment problem

close all;
%clc;


%% Defining nodes to establish all the optimal strategies
[~,~,Centroids] = xlsread('C:\Users\Javier\Desktop\TFM sevilla luis\database
    controll\End database\Nodes\Centroides.xlsx');


for parad = 74
    %% Importing Mathematica information
    tic
    % NODES - Every node used is loaded here
    [~,txt,~] = xlsread('C:\Users\Javier\Desktop\TFM sevilla luis\database
        controll\End database\Nodes\Total_nodes.xls');
    % however this data need to be reorganized. There is no more info needed
    % but in this case we only are going to use the TXT info
    [Unico,ca,~] = unique(txt(:,1)); % elimination of ijntersecting nodes
    % using the same values positionet at the same row
    var = txt(:,2);
    var2 = txt(:,3);
    var3 = txt(:,4);

    var = var(ca);
```

```
var2 = var2(ca);
var3 = var3(ca);

nodes = cell2struct([var var2 var3],Unico,1);
nodes = nodes';

% filling with adequate info
Rnod = Centroids{parad,1}; % destination node
fild = fieldnames(nodes);
for i = 1:length(fild)
    if strcmp(fild{i},Rnod)
        nodes(1).(sprintf('%s',fild{i})) = 0;% using sprintf we can access
            to the data using the name
        nodes(2).(sprintf('%s',fild{i})) = 0;
        disp('tienes nodo')
    else
        nodes(1).(sprintf('%s',fild{i})) = inf;
        nodes(2).(sprintf('%s',fild{i})) = 0;
    end
end

%struct2table(nodes)

clear var var2 var3 num txt raw Unico

% LINKS
[num,txt,raw] = xlsread('C:\Users\Javier\Desktop\TFM sevilla luis\database
    controll\End database\Links\Total_links_end.xlsx');

[Unico,ca,ci] = unique(txt(:,1)); % elimination of ijntersecting nodes
% using the same values positionet at the same row
var = num2cell(num(:,1));
var2 = num(:,2);

% treament for POSSIBLE Nan
indisess = isnan(var2);

var2(indisess) = Inf;
var2 = num2cell(var2);


var3 = txt(:,4);

var = var(ca);
var2 = var2(ca);
var3 = var3(ca);

links = cell2struct([var var2 var3],Unico,1);

links = links';


% removing field links = rmfield(links,'A_B')

S = links;

% 1.2 Get the next link
```

```
g = 1;
% used_nodes{1} = Rnod;
% un = 2;
iter = 1;
reverse_ord = cell(1,length(fieldnames(S)));
Abar = cell(1,length(fieldnames(S)));
j=0;
%spmd
while not(isempty(fieldnames(S)))
    % starting the algorithm
    %   a = (i,j); % perte a S
    % find a which satisfies
    %   u_j + c_a <= u_jpr + c_apr; apr = (ipr,jpr);
    %   S = S - a;
    %   % 1.3 Update node label (Falta un trozo)
    %   ui = (fi*ui + fa*(uj+ca))/(fi+fa);
    %   fi = fi + fa;
    %   Abar = Abar + a;
    tips = fieldnames(nodes);
    linkes = fieldnames(S);
    j = 1;
    mat_uj_ca = zeros(length(linkes),3);
    for i = 1:length(linkes)
       %   u_j + c_a
       %Lparts = strsplit(char(linkes(i)),'_');
       %posi = find(~cellfun(@isempty,gonzalez));
       %gonzalez = strfind(tips,Lparts{2});
       Lparts = regexp(char(linkes(i)),'_','split');
       %[gonzalez2] = ismember(tips,Lparts{2});
       gonzalez2 = strcmp(tips,Lparts{2});
       posi = gonzalez2 == 1;

       %uj = ;
       %ca = ;
       %mat_uj_ca(j,:) = [uj+ca, S(2).(sprintf('%s',linkes{i})) , i ];
       mat_uj_ca(i,:) = [nodes(1).(tips{posi}) + S(1).(linkes{i}),S(2).(
           linkes{i}) , i];
    end

%    mat_uj_ca = arrayfun(@funcy,linkes);
%    size(mat_uj_ca)

    [m,indss] = min(mat_uj_ca(:,1));
    % getting node time info
    %Lparts = strsplit(char(linkes(mat_uj_ca(indss,3))),'_');
    Lparts = regexp(char(linkes(mat_uj_ca(indss,3))),'_','split');
    %[gonzalez2] = ismember(tips,Lparts{1});
    gonzalez2 = strcmp(tips,Lparts{1});
    posi = gonzalez2 == 1;
    %posi = find(~cellfun(@isempty,strfind(tips,Lparts{1})));
    ui = nodes(1).(tips{posi});
    fi = nodes(2).(tips{posi});
    reverse_ord{iter} = linkes(mat_uj_ca(indss,3));
    % eliminating link
    S = rmfield(S,linkes(mat_uj_ca(indss,3)));

    %disp('iteration')
```

```
        % struct2table(nodes)
        % checking if it belongs to the optimal strategy
        if ui >=( mat_uj_ca(indss,1))
            % cheking if infinite value is involved
            fa =  mat_uj_ca(indss,2);
            if isinf(fi) || g==1 || isinf(fa)
               nodes(1).(tips{posi}) = mat_uj_ca(indss,1);
               nodes(2).(tips{posi}) = inf;
            elseif isinf(ui) && fi == 0
                nodes(1).(tips{posi}) = (1 + fa*(mat_uj_ca(indss,1)))/(fi+fa);
                nodes(2).(tips{posi}) = fa + fi;
            else
                nodes(1).(tips{posi}) = (ui*fi + fa*(mat_uj_ca(indss,1)))/(fi+fa
                    );
                nodes(2).(tips{posi}) = fa + fi;
            end
            Abar{g} = linkes(mat_uj_ca(indss,3));
            g = g+1;
        end
        % updating
%         if iter==900
%             break
%         end
        valueee = struct2cell(nodes);
        if any(cellfun(@isnan,valueee(:,1)))
            iter
        end
        clear mat_uj_ca
        iter = iter +1;
        % remeber if
    end
    %end
    filename = sprintf('Optimal strategies found for: %s centroid',Centroids{
        parad,2});
    disp(filename)
   % break

%% saving the important information to define the optima strategie
    filename = sprintf('OptStra_Cent_%s.mat',Centroids{parad,2});
    save(filename,'nodes','links','Abar','reverse_ord')

    toc
end
```

Por lo tanto, tras su aplicación se obtienen las 164 rutas óptimas (en 164 archivos .mat) para cada nodo de destino de la red, restando la asignación de los mismos mediante la matriz origen destino proporcionada por el director.

4.4 Asignación de la ciudad de Sevilla

Se utiliza la base de datos descrita en capítulos anteriores para realizar una asignación de pasajeros en función de los datos. Los resultados obtenidos aplicando el algoritmo sobre los datos de la ciudad de Sevilla son analizados y discutidos. Entre las modalidades analizadas se incluyen:

- Lineas de autobuses y centroides considerando el modo walking para accesos a menos de 500 metros.
- Exactamente el mismo ejemplo pero añadiendo el caso de los transbordos entre buses que esté a menos de 200 metros penalizados con un factor de 1.2 en tiempo.

- Extenso y pormenorizado filtrado de la base de datos para evitar, repeticiones, enlaces de valor 0, cambios en los últimos decimales de las coordenadas introducidas.

El código utilizado para la obtención de rutas óptimas queda descrito en 4.4.

Código 4.4 Algoritmo de Spiess - Parte de las rutas óptimas.

```
%% Optimal strategies by Spiess and florian (second main script)
% This software was implemented by Javier Vázquez Peralta
% TFM Máster en diseño avanzado en ingeniería mecánica
% Supervisor: Luis Miguel Romero Pérez
%
clear all
close all
clc
%% 1st Sevilla city information -- OPTIMAL STRATEGIES PART

% after chaging the cd - it is supossed to be in the correct directory
% because it's the second part of the process

names = dir('*.mat');

% names for loading the info in order to assign
nombres = struct2cell(names);
nombres = nombres(1,:);

% index to control material to load
testece = vertcat(regexp(nombres,'\d*','Match'));
testece = vertcat(testece{:});
testeced = zeros(length(testece),1);
for rrr = 1:length(testece)
    testeced(rrr,1) = str2num(testece{rrr});
end
[~,III] = sort(testeced);
nombres = nombres(III);

% reading journeys to the destination nodes
destinationinf = xlsread('MATRices PUBLICO.xlsx',2); % 1 peak hour, 2 dayly
    assignment

% defining the final nodes
% 2.1 Initialization
%V_i = g_i; % para todo i en I
% reverse_ord must be used in inverse order

 % that is spiess data structure. They must contain (waiting from i to j, and
     frec)

% centroids information
[~,~,Centroids] = xlsread('C:\Users\Javier\Desktop\TFM sevilla luis\database
    controll\End database\Nodes\Centroides_asigg.xlsx');

% initialization of Vi and va

[~,txt,~] = xlsread('C:\Users\Javier\Desktop\TFM sevilla luis\database controll
    \End database\Nodes\Total_nodes.xls');
% however this data need to be reorganized. There is no more info needed
```

```
% but in this case we only are going to use the TXT info
[Unico,~,~] = unique(txt(:,1)); % elimination of ijntersecting nodes
% using the same values positionet at the same row
valss = zeros(length(Unico),1);

Vi = cell(length(Unico),2);
Vi(:,1) = num2cell(valss);
Vi(:,2) = Unico;

%struct2table(nodes)

clear var var2 var3 num txt raw Unico

% LINKS
[num,txt,raw] = xlsread('C:\Users\Javier\Desktop\TFM sevilla luis\database
    controll\End database\Links\Total_links_end.xlsx',3);

[Unico,ca,ci] = unique(txt(:,1)); % elimination of ijntersecting nodes
% using the same values positionet at the same row
valss = zeros(length(Unico),1);
%va = cell2struct(num2cell(valss),Unico,1);

va = cell(length(Unico),2);
va(:,1) = num2cell(valss);
va(:,2) = Unico;

for factor =376 %length(destinationinf)

% 2.2 Loading
tic
    % assign pre gi
    if not(destinationinf(factor,1) == destinationinf(factor,2)) % detec if it
        is repeated
        load(char(nombres(destinationinf(factor,2))))
        %load('OptStra_Cent_88.mat')
        Abar = vertcat(Abar{:});
        Abar=Abar(~cellfun('isempty',Abar'));
        %reverse_ord=reverse_ord(~cellfun('isempty',reverse_ord'));
        % assignment of pre demand
        %jeyjey = strcmp(fieldnames(Vi),Centroids(destinationinf(factor,2),1));
        %any(jeyjey) % detecting destination index
        index = strcmp(Vi(:,2),Centroids(destinationinf(factor,2),1));
        Vi{index,1} = -destinationinf(factor,3); % assign negative for
            destination
        %jeyjey = strcmp(fieldnames(Vi),Centroids(destinationinf(factor,1),1));
        index = strcmp(Vi(:,2),Centroids(destinationinf(factor,1),1));
        Vi{index,1} = destinationinf(factor,3);
        k = 0;
        % Calculating uj + ca matrix
        mat_uj_ca = zeros(length(links(:,1,3)),2);
        link_name = links(:,1,3);
        tips = nodes(:,1,3);
        for i = 1:length(link_name)
            Lparts = regexp(char(link_name(i)),'_','split');
            gonzalez2 = strcmp(tips,Lparts{2}); % another regexp try it
            posi = gonzalez2 == 1;
            mat_uj_ca(i,:) = [nodes{posi,1,1}+links{i,1,1}, links{i,1,2}];
```

```
        end

        % ordering the links taking into account that matrix
        [example,II] = sort(mat_uj_ca(:,1),'descend');
        % ordering links
        link_name = link_name(II);

        for i = 1:length(link_name) % in decreasing order of (u_i + c_a) %%
            REVISAR condicones
            if any(strcmp(link_name(i),Abar))
                Lparts = regexp(char(link_name{i}),'_','split');
                gonzalez1 = strcmp(tips,Lparts{1});
                posi = gonzalez1 == 1;
                % fi
                fi = nodes{posi,1,2};
                % fa
                gonzalezlin = strcmp(links(:,1,3),link_name{i});
                posilin = gonzalezlin == 1;
                fa = links{posilin,1,2};
                %fa = links{i,1,2};
                gonzalez2 = strcmp(tips,Lparts{2});
                posi2 = gonzalez2 == 1;
                if isinf(fa) || isinf(fi)
                    % fi and % fa are inf
                    va{posilin,1} = Vi{posi,1};
                    Vi{posi2,1} = Vi{posi2,1} + va{posilin,1};
                else
                    va{posilin,1} = Vi{posi,1}*(fa)/(fi);
                    Vi{posi2,1} = Vi{posi2,1} + va{posilin,1};
                end

            end
        end
    clear mat_uj_ca
    end
toc
disp(factor)


end


indexi = find([Vi{:,1}] ~= 0);
Vi{indexi,2} % nodes names
index = find([va{:,1}] ~= 0);
vertcat(va{index,2}) % links names
Vi{indexi,1}
Centroids(destinationinf(factor,1),1) % primer nodo
Centroids(destinationinf(factor,2),1) % segundo nodo
```

Puesto que la representación de todas las asignaciones en formato bruto de datos es costoso, confuso y tiene poco contenido para el lector se toman varios ejemplos para ilustrar y demostrar el funcionamiento de la base de datos mostrándolos en detalle. En la tabla 4.1 se muestran estos.

Para el primer caso, se explicará escrito debido a su simplicidad. Siendo las coordenadas de los nodos de origen destino respectivamente *la3740030900000lo599734300000* y *la3739822100000lo599120600000*, se inicializan estas con los valores de volumen en positivo para el nodo de origen y negativo para el nodo de destino como explica *Spiess et al.* en [1]. Tras la aplicación del algoritmo, se obtiene una ruta cuyos enlaces son

Tabla 4.1 Casos escogidos para ejemplificar el comportamiento del algoritmo..

Casos Escogidos	Centroide inicial	Centroide destino	Volumen
1	2	3	305
2	3	9	36
3	3	10	70
4	38	19	45

la3740030900000lo599734300000_la3739702000000lo599439000000 y *la3739702000000lo599439000000_-la3739822100000lo599120600000*. El valor de estos es 305. En el caso de los nodos, ha cambiado para resultar 305 en todos menos en el nodo de destino.

Se utiliza el ejemplo anterior para verificar una de las posibles soluciones que es la ruta única. Pero los 3 casos siguientes contienen varias opciones. Lo cual es la esencia de este algoritmo debido a su tipología explicada en el capítulo 2. Se remarca que estos son casos concretos extraídos de los resultados de algunos de los valores de la matriz OD. Se debe comentar que el modo de representación son líneas que unen puntos en lugar de la ruta completa que seguiría el autobús correspondiente. Se trata de ser ilustrativo mostrando unos resultados que sin esta ayuda sería díficilmente explicables con palabras puesto que estos gráficos le confieren un grado adicional y necesario de visibilidad.

El caso 3 puede ser explicado sin gráfica (al menos no latitud longitud), para los 2 restantes se utilizarán las mismas con el objeto de ser mas ilustrativo, didáctico y claro en las explicaciones. Las coordenadas de los nodos origen destino son *la3739822100000lo599120600000* y *la3739029400000lo599558400000* respectivamente.

El recorrido puede verificarse en 4.1. Aunque todo el recorrido parezca el mismo, en el primer tramo son el mismo, pero en los siguiente se divide el volumen a la mitad para ambos llegando al nodo final por separado. Como se puede apreciar, el algoritmo es capaz de bifurcar la asignación incluso cuando son resultados tan similares como estos en términos de coordenadas. La asignación roja en detalle:

- El primer enlace para ambos: *la3739822100000lo599120600000_la3739702000000lo599439000000* con un número de 70.
- el segundo *la3739702000000lo599439000000_la3739292000000lo599571000000* lleva 35.
- El tercero *la3739292000000lo599571000000_la3739029400000lo599558400000* lleva 35

La asignación marrón en detalle:

- El primer enlace para ambos: *la3739822100000lo599120600000_la3739702000000lo599439000000* con un número de 70.
- el segundo *la3739702000000lo599439000000_la3739314000000lo599570000000* lleva 35.
- El tercero *la3739314000000lo599570000000_la3739029400000lo599558400000* lleva 35

Es obvio que se trata de 2 paradas contiguas por donde pasan varios autobuses, mostrando que este algoritmo no calcula rutas mínimas, sino rutas óptimas.

En el caso 2, los nodos de inicio y destino son *la3739822100000lo599120600000* y *la3738976200000lo600181200000*. En este caso, desde segundo tramo ambas rutas son completamente independientes, se unen únicamente en el nodo de destino. Las rutas del algoritmo pueden comprobarse visualmente en 4.2. Las 2 rutas que se tienen, para la roja:

- El primer enlace que comparten es *la3739822100000lo599120600000_la3739702000000lo599439000000* con un volumen de 36.
- El segundo enlace independiente es *la3739702000000lo599439000000_la3739314000000lo599570000000* con un volumen de 18 (que continúa hasta llegar al nodo destino).
- El tercer enlace independiente es *la3739314000000lo599570000000_la3739029400000lo599558400000*
- El cuarto enlace independiente es *la3739029400000lo599558400000_la3739064000000lo599714000000*
- El quinto enlace independiente es *la3739064000000lo599714000000_la3739029000000lo600086000000*
- El sexto enlace independiente es *la3739029000000lo600086000000_la3738976200000lo600181200000*

En el caso del marrón:

- El primer enlace que comparten es *la3739822100000lo599120600000_la3739702000000lo599439000000* con un volumen de 36.
- El segundo enlace independiente es *la3739702000000lo599439000000_la3739292000000lo599571000000* con un volumen de 18 (que continúa hasta llegar al nodo destino).
- El segundo enlace independiente es *la3739292000000lo599571000000_la3739271000000lo599513000000*
- El segundo enlace independiente es *la3739271000000lo599513000000_la3738046000000lo598588000000*

Se puede apreciar en este caso las rutas diferentes que sigue el algoritmo para asignar diferentes de los casos vistos anteriormente, y en principio, considerando el mapa parece difícil de plantear. Pero se trata de ruta óptima.

En el último caso 4, los nodos de inicio y destino son *la3740794500000lo599861800000* y *la3738656300000lo600603600000*. En este caso, siguen el mismo camino hasta el nodo número 5 donde se separan tomando caminos completamente diferentes para terminar la ruta. En el caso rojo se tiene:

- El primer enlace que comparten es *la3740794500000lo599861800000_la3740448000000lo599527000000* con un volumen de 45.
- El segundo enlace es *la3740448000000lo599527000000_la3740085000000lo599855000000* con un volumen de 45 (que continúa hasta llegar al nodo destino).
- El tercer enlace es *la3740085000000lo599855000000_la3739807000000lo600147000000*
- El cuarto enlace es *la3739807000000lo600147000000_la3739606000000lo600259000000*
- El quinto enlace es *la3739606000000lo600259000000_la3739401000000lo600363000000*
- El sexto enlace independiente es *la3739401000000lo600363000000_la3739063000000lo600423000000* con un volumen de 12.877
- El séptimo enlace independiente es *la3739063000000lo600423000000_la3738656300000lo600603600000* con un volumen de 12.877

En el caso del marrón:

- El primer enlace que comparten es *la3740794500000lo599861800000_la3740448000000lo599527000000* con un volumen de 45.
- El segundo enlace es *la3740448000000lo599527000000_la3740085000000lo599855000000* con un volumen de 45 (que continúa hasta llegar al nodo destino).
- El tercer enlace es *la3740085000000lo599855000000_la3739807000000lo600147000000*
- El cuarto enlace es *la3739807000000lo600147000000_la3739606000000lo600259000000*
- El quinto enlace es *la3739606000000lo600259000000_la3739401000000lo600363000000*
- El sexto enlace independiente es *la3739401000000lo600363000000_la3739169000000lo600344000000* con un volumen de 32.14
- El séptimo enlace independiente es *la3739169000000lo600344000000_la3738761000000lo600220000000* con un volumen de 32.14
- El octavo enlace independiente es *la3738761000000lo600220000000_la3738656300000lo600603600000* con un volumen de 32.14

Aquí se puede apreciar como el reparto entre la bifurcación no es la mitad para cada camino, mostrando por lo tanto las mismas características de cálculo que los resultados del artículo [1]

Finalmente, para obtener la solución de asignación completa para toda la ciudad basta con resolverla para todos los casos de la matriz OD, dependiendo del nodo de destino se utilizará un tipo de ruta óptima. Es decir, con las asignaciones una por una, la suma de todas estas muestra cual es el estado de demanda de los enlances en términos de volumen.

Por otro lado, en palabras de Spiess se podría resolver sin sumar mediante el siguiente cambio, *si la red contiene más de un nodo de destino, el algoritmo se aplica una vez para cada destino, guardando el resultado de rutas óptima, para tras calcular cada uno, asignar utilizando la información previamente guardada*.

Figura 4.1 Rutas de asignación óptima para el caso 3..

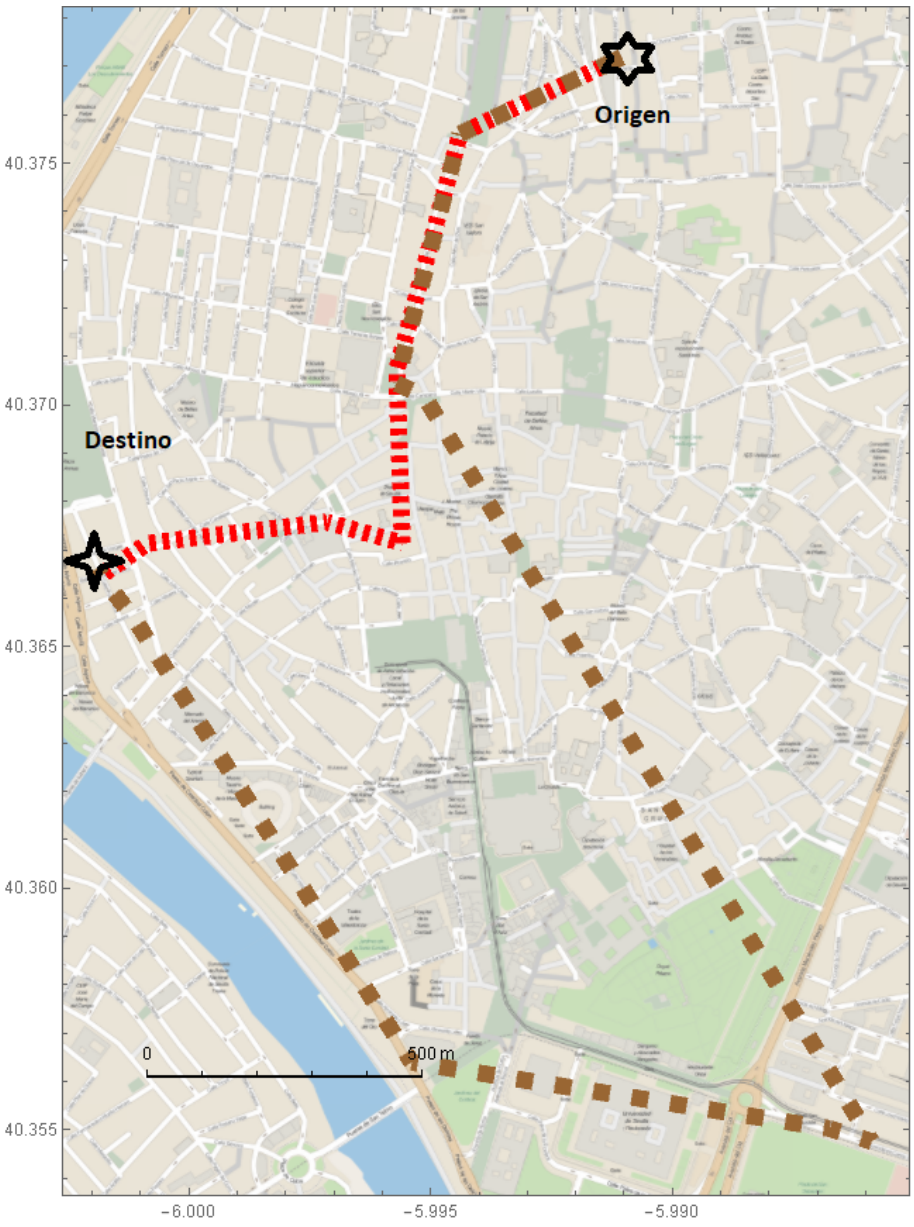

Figura 4.2 Rutas de asignación óptima para el caso 2..

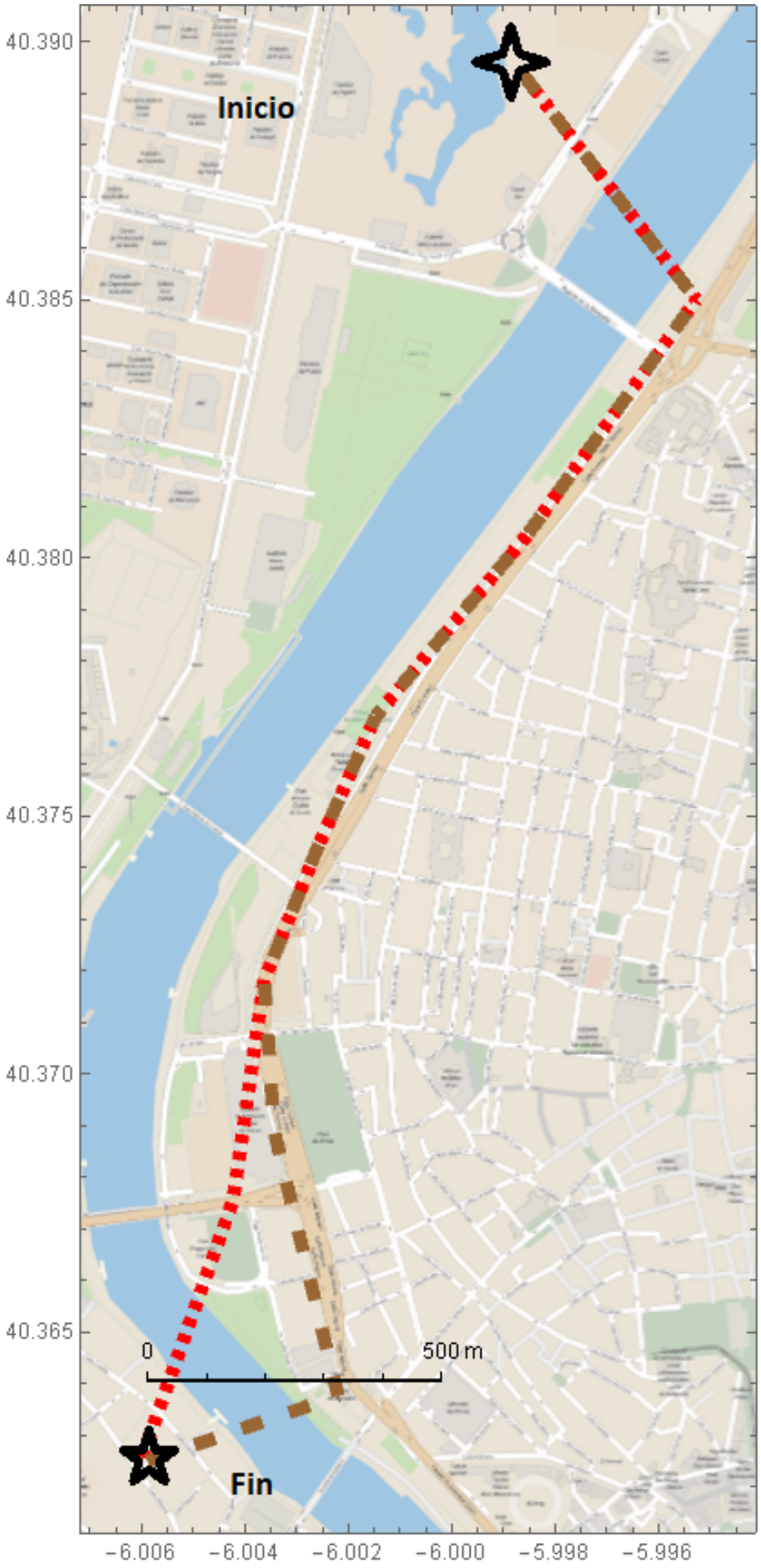

Figura 4.3 Rutas de asignación óptima para el caso 4..

5 Conclusiones Generales

Young man, in mathematics you don't understand things. You just get used to them.

John von Neumann

Se exponen en este apartado las conclusiones obtenidas del estudio, implementación y adaptación del algoritmo son:

5.1 Conclusiones

- Se ha comprobado el funcionamiento correcto del algoritmo, utilizando los datos de la red de Sevilla y del artículo de Spiess como principales fuentes de cotejo de información. Siendo positivo en ambos casos.
- El algoritmo resuelve la asignación completa para la ciudad de Sevilla. Para hacer esto, obtiene las estrategias o rutas óptimas desde cada centroide de la zonificación de Sevilla, que dependiendo de los datos utilizados, podrá ser el caso de hora punta de Sevilla, de diario o durante un fin de semana.
- La asignación obtiene los mismo valores que el artículo de [1] mimetizando los resultados. Siendo esta una garantía correcto funcionamiento.
- Los resultados obtenidos con la red de Sevilla son razonables considerando los caminos y tiempos obtenidos del algoritmo. Por lo tanto se obtiene la misma conclusión que en el punto anterior.
- Se ha resuelto de forma satisfactoria la gestión, obtención, organización y almacenamiento de la base de datos que contiene los datos necesarios de la red de Sevilla. Del mismo modo, la peticiones formuladas, las rutinas para filtrar datos y para organizarlos son correctas. Se han propuesto alternativas diferentes que puedan ser de utilidad para el tutor de este trabajo.
- Cambiar el modelo de reconocimiento de enlaces durante el algoritmo para realizar cotejos con los nombres que *Tussam* le otorga a cada parada con un código interno numérico. Para el caso de los centroides, ya que también son números se propone otorgarles un nombre de esta forma $c125$ donde después de la c vendría el número de centroide asociado.
- Este trabajo no es *friendly user* para no iniciados en programación. Para personas sin conocimientos básicos de programación, de gestión de datos entre programas no se recomienda bajo ningún concepto este enfoque de resolución para esta tarea, salvo que el usuario desee un reto. En este caso, será bajo su responsabilidad ya que en algunas zonas envuelve conceptos complejos de programación como la paralelización de un cálculo.
- Aunque el algoritmo resuelve el problema pedido, considerando los cálculos y el tiempo que tardan, se infiere que puede ser del orden de 10 veces más rápido que actualmente.
- Los datos de la base de datos son lo más completos posibles pero aún de este modo siempre hay factores que escapan al control de autor para reducir el efecto que tengan sobre los datos y resultados.

- Un usuario habitual de *Mathematica* habría tenido unas grandes ventajas en toda la gestión de la base de datos porque las propiedades de la programación funcional son múltiples y ventajosas. Por lo tanto, un programador de *Haskell* que acostumbre a trabajar en un entorno parecido partirá con una gran ventaja.

5.2 Lineas futuras

Como posibles desarrollos y lineas futuras para ampliar y mejorar el algoritmo se proponen:

- Adaptar el algoritmo para extraer los datos de una database mas completa y en formato *GTFS*.
- Crear metodologías de obtención de datos que sean consistentes e independientes dentro de lo posible de otras entidades que limiten el acceso a la información. Aunque no siempre será posible.
- Mejora de la velocidad del algoritmo
- Incluir las lineas de metro dentro del modelo de transporte de la ciudad de Sevilla. Además, ampliar hasta los pueblos mas próximos. Incluir todo el transporte público relevante en el área metropolitana haciendo un recalculado.
- Realizar los algoritmos de extracción de información desde *Google Maps* utilizando *ToAssociations* en *Mathematica* en lugar de utilizar un posicionamiento mediante las funciones *Position* y la creación de expresiones nuevas utilizando *ToString* junto a *ToExpression*.
- Pedir a *Tussam* sus bases de datos para dotar al proyecto de una carácter mas realista.
- Aplicar la misma metodología de extracción de datos para un algoritmo de transporte privado visto en el capítulo 2.
- Realizar un programa sincronizado desde el punto de vista que los cálculos puedan ser lanzados a tiempo real. Esto requeriría de una colaboración de carácter completo con la empresa *Google* o alguna otra que proporcione datos de esta forma.
- No suponer promedios de datos, incluir la opción de calcular según la franja horaria y la época del año.
- Programar la versión no lineal (que se encuentra en [1]) para establecer una comparativa entre las 2 versiones que proporciona Spiess.
- Establecer unos nuevos cálculos basados en la población desplazando los centroides en función de la densidad de población de las zonas de Sevilla.
- Establecer una comparativa de resultados con el programa *EMME/2* u otro software comercial.

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] H. Spiess and M. Florian, "Optimal strategies: A new assignment model for transit networks," *Transportation Research Part B: Methodological*, vol. 23, no. 2, pp. 83 – 102, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0191261589900349

[2] J. de Dios Ortuzar and L. Willumsen, *Modelling Transport*. Wiley, 2011. [Online]. Available: https://books.google.es/books?id=qWa5MyS4CiwC

[3] G. Gentile and K. Noekel, *Modelling Public Transport Passenger Flows in the Era of Intelligent Transport Systems: COST Action TU1004 (TransITS)*, ser. Springer Tracts on Transportation and Traffic. Springer International Publishing, 2016. [Online]. Available: https://books.google.es/books?id=DDmFCwAAQBAJ

[4] P. Marcotte and S. Nguyen, *Equilibrium and Advanced Transportation Modelling*, ser. Centre for Research on Transportation. Springer US, 2013. [Online]. Available: https://books.google.com.na/books?id=6fXlBwAAQBAJ

[5] Q. Fu, R. Liu, and S. Hess, "A review on transit assignment modelling approaches to congested networks: A new perspective," *Procedia - Social and Behavioral Sciences*, vol. 54, pp. 1145 – 1155, 2012, proceedings of EWGT2012 - 15th Meeting of the EURO Working Group on Transportation, September 2012, Paris. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877042812042917

[6] K. . Associates, U. S. F. T. Administration, T. C. R. Program, T. D. Corporation, and N. R. C. U. T. R. Board, *Transit Capacity and Quality of Service Manual*, ser. Report (Transit Cooperative Research Program). Transportation Research Board, 2003. [Online]. Available: https://books.google.es/books?id=Ddf9pqTONhwC

[7] H. Spiess and U. de Montréal. Centre de recherche sur les transports, *Contributions à la théorie et aux outils de planification des réseaux de transport urbain*, ser. Publication (Université de Montréal. Centre de recherche sur les transports). Université de Montréal, Département dínformatique et de recherche opérationnelle - Faculté des arts et des sciences, 1984. [Online]. Available: https://books.google.es/books?id=n8kYuAAACAAJ

[8] Google, "Google apis," Jun. 2017. [Online]. Available: https://developers.google.com/maps/web-services/

[9] J. Devore, *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 2015. [Online]. Available: https://books.google.es/books?id=zzV-BAAAQBAJ

[10] M. Developers, "Javascript guide," Aug. 2017. [Online]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript/Guide

[11] D. Flanagan, *JavaScript: The Definitive Guide: The Definitive Guide*. O'Reilly Media, 2006. [Online]. Available: https://books.google.es/books?id=2weL0iAfrEMC

[12] Google, "Javascript peticiones," Aug. 2017. [Online]. Available: https://developers.google.com/maps/documentation/javascript/?hl=es-419

[13] M. Abell and J. Braselton, *The Mathematica Handbook*. Elsevier Science, 2014. [Online]. Available: https://books.google.es/books?id=96DiBQAAQBAJ

[14] MathWorks, "Mathematica documentation center," 2017. [Online]. Available: http://reference.wolfram.com/language/

[15] J. Borwein and M. Skerritt, *An Introduction to Modern Mathematical Computing: With MapleTM*, ser. Springer Undergraduate Texts in Mathematics and Technology. Springer New York, 2011. [Online]. Available: https://books.google.es/books?id=4Kp9PbSAR4IC

[16] W. R. (Firm), *Wolfram Mathematica Tutorial Collection: Data Manipulation*, ser. Wolfram Mathematica Tutorial Collection. Wolfram Research, Incorporated, 2008. [Online]. Available: https://books.google.es/books?id=nyChYgEACAAJ

[17] Mathematica, *DATABASELINK USER GUIDE*, Wolfram Research Inc., 2008, http://reference.wolfram.com/language/DatabaseLink/tutorial/Overview.html.

[18] ——, *Jlink User Guide*, Wolfram Research Inc., 2008, http://reference.wolfram.com/language/JLink/tutorial/Overview.html.

[19] D. Xue and Y. Chen, *Scientific Computing with MATLAB, Second Edition*. CRC Press, 2016. [Online]. Available: https://books.google.es/books?id=Ep6mCwAAQBAJ

[20] Google, "Places api," Apr. 2017. [Online]. Available: https://developers.google.com/places/web-service/intro?hl=es-419

[21] ——, "Directions api," Aug. 2017. [Online]. Available: https://developers.google.com/maps/documentation/directions/start?hl=es-419

[22] A. Beaulieu, *Learning SQL*. O'Reilly Media, 2005. [Online]. Available: https://books.google.es/books?id=OQr4DvaPsLwC

[23] A. Wilson, *Entropy in Urban and Regional Modelling*, ser. Monographs in spatial and environmental systems analysis. Routledge, 2011. [Online]. Available: https://books.google.es/books?id=0HTKq7GHZ4UC

[24] F. H. Knight, "Some fallacies in the interpretation of social cost," *The Quarterly Journal of Economics*, vol. 38, no. 4, pp. 582–606, 1924. [Online]. Available: +http://dx.doi.org/10.2307/1884592

[25] J. Wardrop, *Some Theoretical Aspects of Road Traffic Research*, ser. Road paper. Institution of Civil Engineers, 1952. [Online]. Available: https://books.google.es/books?id=9zEpAQAAMAAJ

[26] M. Beckmann, C. McGuire, and C. Winsten, *Studies in the Economics of Transportation*. Yale University Press, 1956. [Online]. Available: https://books.google.es/books?id=3CVPAAAAMAAJ

[27] D. Braess, "Über ein paradoxon aus der verkehrsplanung," *Unternehmensforschung*, vol. 12, no. 1, pp. 258–268, Dec 1968. [Online]. Available: https://doi.org/10.1007/BF01918335

[28] M. J. Mogridge, "The self-defeating nature of urban road capacity policy: A review of theories, disputes and available evidence," *Transport Policy*, vol. 4, no. 1, pp. 5 – 23, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967070X96000303

[29] M. Mogridge, *Travel in Towns: Jam Yesterday, Jam Today and Jam Tomorrow?*, ser. Macmillan reference books. MacMillan, 1990. [Online]. Available: https://books.google.es/books?id=FyVPAAAAMAAJ

[30] J. Cea, J. Larrañaga, and U. C. de Chile. Departamento de Transporte, *Transit Assignment to Minimal Routes: An Efficient New Algorithm*, ser. Documento de trabajo. Universidad católica de Chile, Departamento de transporte, 1988. [Online]. Available: https://books.google.es/books?id=tHMiAQAAMAAJ

[31] Google, "Google query example," Jun. 2017. [Online]. Available: https://developers.google.com/maps/web-services/overview

[32] ——, “Google api ejemplo de petición,” Jun. 2017. [Online]. Available: https://developers.google.com/maps/documentation/directions/start

[33] Tussam, “Web inicial de la empresa de transportes sevillana,” Dec. 2016. [Online]. Available: http://www.tussam.es/index.php?id=1

[34] Google, “Ejemplo de petición,” Jun. 2017. [Online]. Available: https://developers.google.com/maps/web-services/overview?hl=es-419

[35] ——, “Google optimization tools,” Aug. 2017. [Online]. Available: https://developers.google.com/optimization/

[36] ——, “Google optimization tools github,” Aug. 2017. [Online]. Available: https://github.com/google/or-tools

[37] ——, “Get started: Key,” Aug. 2017. [Online]. Available: https://developers.google.com/maps/documentation/directions/get-api-key?hl=es-419

[38] ——, “Key good use reccomendations,” Aug. 2017. [Online]. Available: https://support.google.com/googleapi/answer/6310037?hl=es-419

[39] ——, “Account limitations,” Aug. 2017. [Online]. Available: https://developers.google.com/maps/documentation/directions/usage-limits?hl=es-419

[40] ——, “Directions queries information,” Aug. 2017. [Online]. Available: https://developers.google.com/maps/documentation/directions/intro?hl=es-419

[41] P. Wellin, *Programming with Mathematica®: An Introduction*. Cambridge University Press, 2013. [Online]. Available: https://books.google.es/books?id=-mgQGZTSsCYC

[42] J. Fernández Tejada, “Evaluación de la accesibilidad al transporte público en sevilla mediante simulaciones macroscópicas,” 2016.

[43] A. de Sevilla, “Encuesta demanda de movilidad de 2007,” 2007.